**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**pulsador-lento-rapido-con-servo.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

entity main is

port (

clk : in std\_logic;

sw : in STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- interruptores

-- btnU : in STD\_LOGIC; -- boton arriba

-- btnD : in STD\_LOGIC; -- boton abajo

btnL : in STD\_LOGIC; -- boton izquierda

btnR : in STD\_LOGIC; -- boton derecha

btnC : in STD\_LOGIC; -- boton central

led : out STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- leds

seg : out STD\_LOGIC\_VECTOR(6 DOWNTO 0); -- siete segmentos

dp : out STD\_LOGIC; -- punto decimal del siete segmentos

an : out STD\_LOGIC\_VECTOR(3 DOWNTO 0); -- control de 7-seg

servo : out std\_logic

);

end main;

architecture Behavioral of main is

-- signals del servomotor

signal estado\_servo: std\_logic\_vector (1 downto 0);

signal selector\_aspersor\_mode: std\_logic; -- sw(11)

signal selector\_input\_mode: std\_logic; -- sw(10)

signal aspersor\_cont: std\_logic\_vector(3 downto 0);

signal selector\_switches: std\_logic\_vector (3 downto 0); -- sw(3 downto 0)

signal grados: integer range 0 to 180;

signal cont\_flancos: integer range 0 to 2000000; -- (20 ms) -> (50 Hz)

signal pwm\_longitud\_pulso: integer range 0 to 2000000; -- (20 ms) -> (50 Hz)

-- signals servomotor-reloj

signal segundos\_offset: std\_logic\_vector(3 downto 0); -- sw(15 downto 12)

signal suma\_o\_resta: std\_logic := '0';

signal cont\_base: integer range 0 to 400000000; -- lleva la cuenta del reloj, puesto para 1-4 seg...

signal tope\_freq: integer range 0 to 400000000;

-- signals del pulsador

signal estado\_pulsador: std\_logic\_vector (2 downto 0);

signal cont\_filtro: integer range 0 to 500000000;

signal salida: std\_logic;

signal flag\_suma: std\_logic;

signal flag\_resta: std\_logic;

signal freq\_min: integer range 0 to 100000000;

signal contador\_centenas: std\_logic\_vector (3 downto 0);

signal contador\_decenas: std\_logic\_vector (3 downto 0);

signal contador\_base\_enable: integer range 0 to 100000;

signal enable\_seg\_aux: std\_logic\_vector (3 downto 0);

signal dato: std\_logic\_vector (3 downto 0);

signal btnU: std\_logic; -- sw(5)

signal btnD: std\_logic; -- sw(4)

-- signals pulsador-pwm

signal contador\_decenas\_integer: integer range 0 to 9;

signal contador\_centenas\_integer: integer range 0 to 9;

signal numero\_int: integer range 0 to 200;

begin

-- ####################################################################

-- ####################################################################

-- LOGICA DEL SERVOMOTOR

-- ####################################################################

-- ####################################################################

segundos\_offset <= sw(15 downto 12);

selector\_aspersor\_mode <= sw(11);

selector\_input\_mode <= sw(10);

selector\_switches <= sw(3 downto 0);

-- process de designacià¸£à¸“n de grados por tiempo/switches/dedo

process(selector\_aspersor\_mode, aspersor\_cont, selector\_input\_mode, selector\_switches, suma\_o\_resta, numero\_int)

begin

-- grados por aspersor

if selector\_aspersor\_mode = '1' then

case aspersor\_cont is

when "0000" => grados <= 10;

when "0001" => grados <= 20;

when "0010" => grados <= 30;

when "0011" => grados <= 40;

when "0100" => grados <= 50;

when "0101" => grados <= 60;

when "0110" => grados <= 70;

when "0111" => grados <= 80;

when "1000" => grados <= 90;

when "1001" => grados <= 100;

when "1010" => grados <= 110;

when "1011" => grados <= 120;

when "1100" => grados <= 130;

when "1101" => grados <= 140;

when "1110" => grados <= 150;

when "1111" => grados <= 170;

when others => grados <= 10;

end case;

led(3 downto 0) <= aspersor\_cont;

led(8) <= suma\_o\_resta;

-- grados por switches

elsif selector\_input\_mode = '0' then

case selector\_switches is

when "0000" => grados <= 10;

when "0001" => grados <= 20;

when "0010" => grados <= 30;

when "0011" => grados <= 40;

when "0100" => grados <= 50;

when "0101" => grados <= 60;

when "0110" => grados <= 70;

when "0111" => grados <= 80;

when "1000" => grados <= 90;

when "1001" => grados <= 100;

when "1010" => grados <= 110;

when "1011" => grados <= 120;

when "1100" => grados <= 130;

when "1101" => grados <= 140;

when "1110" => grados <= 150;

when "1111" => grados <= 170;

when others => grados <= 10;

end case;

led(3 downto 0) <= "0000";

led(8) <= '0';

-- grados por dedo

else

grados <= numero\_int;

led(3 downto 0) <= "0000";

led(8) <= '0';

end if;

end process;

pwm\_longitud\_pulso <= grados \* 1111 + 50000;

-- process del automata del pwm del servo

process(clk, btnC)

begin

if btnC = '1' then

estado\_servo <= "00";

cont\_flancos <= 0;

elsif rising\_edge(clk) then

case estado\_servo is

when "00" =>

cont\_flancos <= 0;

estado\_servo <= "01";

when "01" =>

cont\_flancos <= 1;

estado\_servo <= "10";

when "10" =>

cont\_flancos <= cont\_flancos + 1;

if cont\_flancos = pwm\_longitud\_pulso then

estado\_servo <= "11";

else

estado\_servo <= "10";

end if;

when "11" =>

cont\_flancos <= cont\_flancos + 1;

if cont\_flancos = 2000000 then

estado\_servo <= "01";

else

estado\_servo <= "11";

end if;

when others =>

cont\_flancos <= 0;

estado\_servo <= "00";

end case;

end if;

end process;

-- process de salidas del servo

process(estado\_servo)

begin

case estado\_servo is

when "00" => servo <= '0';

when "01" => servo <= '1';

when "10" => servo <= '1';

when "11" => servo <= '0';

when others => servo <= '0';

end case;

end process;

-- ####################################################################

-- ####################################################################

-- LOGICA DEL RELOJ

-- ####################################################################

-- ####################################################################

-- proceso de reloj

process(btnC, clk)

begin

if btnC = '1' then

cont\_base <= 0;

elsif rising\_edge(clk) then

if cont\_base = tope\_freq then

cont\_base <= 0;

else

cont\_base <= cont\_base + 1;

end if;

end if;

end process;

-- process de cambio de vel.

process(segundos\_offset)

begin

if segundos\_offset = "XXX1" then

led(15 downto 12) <= "0001";

tope\_freq <= 100000000;

elsif segundos\_offset = "XX10" then

led(15 downto 12) <= "0010";

tope\_freq <= 200000000;

elsif segundos\_offset = "X100" then

led(15 downto 12) <= "0100";

tope\_freq <= 300000000;

elsif segundos\_offset = "1000" then

led(15 downto 12) <= "1000";

tope\_freq <= 400000000;

else

led(15 downto 12) <= "0000";

tope\_freq <= 100000000;

end if;

end process;

-- process de cambio de aspersor\_cont

process(btnC, clk)

begin

if btnC = '1' then

suma\_o\_resta <= '0';

aspersor\_cont <= "0000";

elsif rising\_edge(clk) then

if selector\_aspersor\_mode = '1' then

if cont\_base = tope\_freq then

if aspersor\_cont = "1111" then

suma\_o\_resta <= '1';

elsif aspersor\_cont = "0000" then

suma\_o\_resta <= '0';

end if;

if suma\_o\_resta = '0' and aspersor\_cont /= "1111" then

aspersor\_cont <= aspersor\_cont + 1;

elsif suma\_o\_resta = '1' and aspersor\_cont /= "0000" then

aspersor\_cont <= aspersor\_cont - 1;

end if;

end if;

else

suma\_o\_resta <= '0';

aspersor\_cont <= "0000";

end if;

end if;

end process;

-- ####################################################################

-- ####################################################################

-- LOGICA DEL PULSADOR

-- ####################################################################

-- ####################################################################

btnU <= sw(5);

btnD <= sw(4);

freq\_min <= 100000;

-- process del automata pulsador dedo

process(btnC, clk)

begin

if btnC = '1' then

estado\_pulsador <= "000";

cont\_filtro <= 0;

elsif rising\_edge(clk) then

case estado\_pulsador is

when "000" => -- INICIO

cont\_filtro <= 0;

if btnU = '1' or btnD = '1' then

estado\_pulsador <= "001";

else

estado\_pulsador <= "000";

end if;

when "001" => -- FILTRADO

cont\_filtro <= cont\_filtro + 1;

if (btnU = '1' or btnD = '1') and cont\_filtro < freq\_min then

estado\_pulsador <= "001";

elsif (btnU = '1' or btnD = '1') and cont\_filtro = freq\_min then

if btnU = '1'then

estado\_pulsador <= "010";

elsif btnD = '1' then

estado\_pulsador <= "100";

end if;

else

estado\_pulsador <= "000";

end if;

when "010" => -- UNO +

cont\_filtro <= cont\_filtro + 1;

if btnU = '1' and cont\_filtro < 200000000 then

estado\_pulsador <= "010";

elsif btnU = '1' and cont\_filtro = 200000000 then

estado\_pulsador <= "110";

elsif btnU = '0' then

estado\_pulsador <= "011";

end if;

when "011" => -- SUMA

cont\_filtro <= 0;

if btnU = '1' then

estado\_pulsador <= "001";

else

estado\_pulsador <= "000";

end if;

when "100" => -- UNO -

cont\_filtro <= cont\_filtro + 1;

if btnD = '1' and cont\_filtro < 200000000 then

estado\_pulsador <= "100";

elsif btnD = '1' and cont\_filtro = 200000000 then

estado\_pulsador <= "110";

elsif btnD = '0' then

estado\_pulsador <= "101";

end if;

when "101" => -- RESTA

cont\_filtro <= 0;

if btnD = '1' then

estado\_pulsador <= "001";

else

estado\_pulsador <= "000";

end if;

when "110" => -- START\_RAPIDO

cont\_filtro <= 0;

if btnU = '1' or btnD = '1' then

estado\_pulsador <= "111";

elsif btnU = '0' and btnD = '0' then

estado\_pulsador <= "000";

end if;

when "111" => -- RAPIDO

cont\_filtro <= cont\_filtro + 1;

if (btnU = '1' or btnD = '1') and cont\_filtro < 20000000 then

estado\_pulsador <= "111";

elsif (btnU = '1' or btnD = '1') and cont\_filtro = 20000000 then

estado\_pulsador <= "110";

elsif btnU = '0' and btnD = '0' then

estado\_pulsador <= "000";

end if;

when others =>

cont\_filtro <= 0;

estado\_pulsador <= "000";

end case;

end if;

end process;

-- process de las salidas pulsador dedo

process(estado\_pulsador)

begin

case estado\_pulsador is

when "000" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "001" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "010" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "011" =>

salida <= '1';

flag\_suma <= '1';

flag\_resta <= '0';

when "100" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "101" =>

salida <= '1';

flag\_suma <= '0';

flag\_resta <= '1';

when "110" =>

salida <= '1';

if btnU = '1' then

flag\_suma <= '1';

flag\_resta <= '0';

elsif btnD = '1' then

flag\_suma <= '0';

flag\_resta <= '1';

end if;

when "111" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when others =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

end case;

end process;

-- process de sumar/restar decenas

process(btnC, clk)

begin

if btnC = '1' then

contador\_decenas <= "0001";

elsif rising\_edge(clk) then

if salida = '1' then

if flag\_suma = '1' then

if contador\_decenas = 7 and contador\_centenas = 1 then

contador\_decenas <= "0111";

elsif contador\_decenas = 9 then

contador\_decenas <= "0000";

else

contador\_decenas <= contador\_decenas + 1;

end if;

elsif flag\_resta = '1' then

if contador\_decenas = 1 and contador\_centenas = 0 then

contador\_decenas <= "0001";

elsif contador\_decenas = 0 then

contador\_decenas <= "1001";

else

contador\_decenas <= contador\_decenas - 1;

end if;

end if;

end if;

end if;

end process;

-- process de sumar/restar centenas

process(btnC, clk)

begin

if btnC = '1' then

contador\_centenas <= "0000";

elsif rising\_edge(clk) then

if salida = '1' then

if flag\_suma = '1' then

if contador\_decenas = 9 then

contador\_centenas <= contador\_centenas + 1;

end if;

elsif flag\_resta = '1' then

if contador\_centenas = 1 and contador\_decenas = 0 then

contador\_centenas <= contador\_centenas - 1;

end if;

end if;

end if;

end if;

end process;

contador\_decenas\_integer <= conv\_integer(contador\_decenas);

contador\_centenas\_integer <= conv\_integer(contador\_centenas);

numero\_int <= ((contador\_centenas\_integer \* 10) + contador\_decenas\_integer) \* 10;

-- proceso de frecuencia para el control del enable\_seg\_aux

process(clk, btnC)

begin

if btnC = '1' then

contador\_base\_enable <= 0;

elsif rising\_edge(clk) then

if contador\_base\_enable = 100000 then

contador\_base\_enable <= 0;

else

contador\_base\_enable <= contador\_base\_enable + 1;

end if;

end if;

end process;

-- proceso de control del enable\_seg\_aux

process(clk, btnC)

begin

if btnC = '1' then

enable\_seg\_aux <= "0111";

elsif rising\_edge(clk) then

if contador\_base\_enable = 100000 then

enable\_seg\_aux <= enable\_seg\_aux(2 downto 0) & enable\_seg\_aux(3);

end if;

end if;

end process;

an <= enable\_seg\_aux;

-- proceso de display de diferentes valores en diferentes siete\_segs

process(enable\_seg\_aux, contador\_decenas, contador\_centenas)

begin

if grados < 100 then

case enable\_seg\_aux is

when "0111" => dato <= "1111";

when "1011" => dato <= std\_logic\_vector(to\_unsigned(grados / 100, 4));

when "1101" => dato <= std\_logic\_vector(to\_unsigned(grados / 10, 4));

when "1110" => dato <= "0000";

when others => dato <= "1111";

end case;

else

case enable\_seg\_aux is

when "0111" => dato <= "1111";

when "1011" => dato <= std\_logic\_vector(to\_unsigned(grados / 100, 4));

when "1101" => dato <= std\_logic\_vector(to\_unsigned((grados / 10) - 10, 4));

when "1110" => dato <= "0000";

when others => dato <= "1111";

end case;

end if;

end process;

-- proceso de display de diferentes valores en diferentes siete\_segs

process(dato)

begin

case dato is

when "0000" => seg <= "0000001";

when "0001" => seg <= "1001111";

when "0010" => seg <= "0010010";

when "0011" => seg <= "0000110";

when "0100" => seg <= "1001100";

when "0101" => seg <= "0100100";

when "0110" => seg <= "1100000";

when "0111" => seg <= "0001111";

when "1000" => seg <= "0000000";

when "1001" => seg <= "0001100";

when others => seg <= "1111111";

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**conversor-binario-bd.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

entity main is

port (

clk : in std\_logic;

sw : in STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- interruptores

btnU : in STD\_LOGIC; -- boton arriba

btnD : in STD\_LOGIC; -- boton abajo

btnL : in STD\_LOGIC; -- boton izquierda

btnR : in STD\_LOGIC; -- boton derecha

btnC : in STD\_LOGIC; -- boton central

led : out STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- leds

seg : out STD\_LOGIC\_VECTOR(6 DOWNTO 0); -- siete seg

dp : out STD\_LOGIC; -- punto decimal del siete seg

an : out STD\_LOGIC\_VECTOR(3 DOWNTO 0); -- control de 7-seg

dcmotor : out std\_logic\_vector (1 downto 0);

servo : out std\_logic

);

end main;

architecture Behavioral of main is

-- signals de control

signal inicio: std\_logic;

signal binario: std\_logic\_vector (3 downto 0);

signal enable: std\_logic;

signal fin: std\_logic;

-- signals de conversion

signal estado\_conversion: std\_logic\_vector (1 downto 0);

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

signal contador\_desplazamientos: integer range 0 to 7;

signal unidades: std\_logic\_vector (3 downto 0);

signal decenas: std\_logic\_vector (3 downto 0);

-- signals del reloj

signal cont\_base\_enable: integer range 0 to 100000;

signal cont: integer range 0 to 100000000;

signal tope\_freq: integer range 0 to 400000000;

signal modo\_lento\_rapido: std\_logic;

-- signals de siete-segmentos

signal sal\_mux: std\_logic\_vector (3 downto 0);

signal enable\_seg: std\_logic\_vector (3 downto 0);

signal segmentos: std\_logic\_vector (6 downto 0);

begin

inicio <= btnC;

binario <= sw(3 downto 0);

enable <= sw(15);

modo\_lento\_rapido <= sw(14);

led(15) <= fin;

led(14) <= modo\_lento\_rapido;

led(11 downto 0) <= vector;

-- ####################################################################

-- ####################################################################

-- LOGICA DE CONVERSION

-- ####################################################################

-- ####################################################################

-- process del automata de la conversion

process(clk, inicio)

begin

if inicio = '1' then

vector <= "000000000000";

estado\_conversion <= "00";

contador\_desplazamientos <= 0;

unidades <= "0000";

decenas <= "0000";

fin <= '0';

elsif rising\_edge(clk) then

if cont = 0 and fin = '0' then

case estado\_conversion is

-- start

when "00" =>

contador\_desplazamientos <= 0;

vector <= "00000000" & binario;

if enable = '1' or btnU = '1' then

estado\_conversion <= "01";

else

estado\_conversion <= "00";

end if;

fin <= '0';

-- despl

when "01" =>

contador\_desplazamientos <= contador\_desplazamientos + 1;

vector <= vector(10 downto 0) & '0';

if contador\_desplazamientos < 3 then

estado\_conversion <= "10";

else

estado\_conversion <= "11";

end if;

fin <= '0';

-- ¿sumar+3?

when "10" =>

contador\_desplazamientos <= contador\_desplazamientos;

if vector(11 downto 8) > 4 then

vector(11 downto 8) <= vector(11 downto 8) + "0011";

end if;

if vector(7 downto 4) > 4 then

vector(7 downto 4) <= vector(7 downto 4) + "0011";

end if;

estado\_conversion <= "01";

fin <= '0';

-- final

when "11" =>

contador\_desplazamientos <= contador\_desplazamientos;

vector <= vector;

estado\_conversion <= "00";

fin <= '1';

unidades <= vector(7 downto 4);

decenas <= vector(11 downto 8);

when others =>

contador\_desplazamientos <= 0;

vector <= "000000000000";

estado\_conversion <= "00";

fin <= '0';

unidades <= "0000";

decenas <= "0000";

end case;

end if;

end if;

end process;

-- ####################################################################

-- ####################################################################

-- LOGICA DEL RELOJ

-- ####################################################################

-- ####################################################################

-- process de conteo de segundos

process(clk, inicio)

begin

if inicio = '1' then

cont <= 0;

elsif rising\_edge(clk) then

if cont = tope\_freq then

cont <= 0;

else

cont <= cont + 1;

end if;

end if;

end process;

-- process de cambio de vel.

process(modo\_lento\_rapido)

begin

if modo\_lento\_rapido = '1' then

tope\_freq <= 0;

else

tope\_freq <= 50000000;

end if;

end process;

-- ####################################################################

-- ####################################################################

-- LOGICA DEL 7SEG

-- ####################################################################

-- ####################################################################

an <= enable\_seg;

seg <= segmentos;

-- process de conteo de freq para multiplex del siete-segmentos

process(inicio, clk)

begin

if inicio = '1' then

cont\_base\_enable <= 0;

elsif rising\_edge(clk) then

if cont\_base\_enable = 100000 then

cont\_base\_enable <= 0;

else

cont\_base\_enable <= cont\_base\_enable + 1;

end if;

end if;

end process;

-- process de multiplexado del siete-segmentos

process(clk,inicio)

begin

if inicio = '1' then

enable\_seg <= "1110";

elsif rising\_edge(clk) then

if cont\_base\_enable = 100000 then

enable\_seg <= enable\_seg(2 downto 0) & enable\_seg(3);

end if;

end if;

end process;

--process de multiplexado de las entradas al 7-seg

process(enable\_seg, unidades, decenas)

begin

case enable\_seg is

when "0111" => sal\_mux <= "0000";

when "1011" => sal\_mux <= "0000";

when "1101" => sal\_mux <= decenas;

when "1110" => sal\_mux <= unidades;

when others => sal\_mux <= "0000";

end case;

end process;

-- process de salidas al siete-segmentos

process(sal\_mux)

begin

case sal\_mux is

when "0000" => segmentos <= "0000001";

when "0001" => segmentos <= "1001111";

when "0010" => segmentos <= "0010010";

when "0011" => segmentos <= "0000110";

when "0100" => segmentos <= "1001100";

when "0101" => segmentos <= "0100100";

when "0110" => segmentos <= "1100000";

when "0111" => segmentos <= "0001111";

when "1000" => segmentos <= "0000000";

when "1001" => segmentos <= "0001100";

when others => segmentos <= "1111111";

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**servo-completo.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

entity main is

port (

clk : in std\_logic;

sw : in STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- interruptores

btnU : in STD\_LOGIC; -- boton arriba

btnD : in STD\_LOGIC; -- boton abajo

btnL : in STD\_LOGIC; -- boton izquierda

btnR : in STD\_LOGIC; -- boton derecha

btnC : in STD\_LOGIC; -- boton central

led : out STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- leds

seg : out STD\_LOGIC\_VECTOR(6 DOWNTO 0); -- siete segmentos

dp : out STD\_LOGIC; -- punto decimal del siete segmentos

an : out STD\_LOGIC\_VECTOR(3 DOWNTO 0); -- control de 7-seg

servo : out std\_logic

);

end main;

architecture Behavioral of main is

-- signals del servomotor

signal estado\_servo: std\_logic\_vector (1 downto 0);

signal selector\_aspersor\_mode: std\_logic; -- sw(11)

signal selector\_input\_mode: std\_logic; -- sw(10)

signal aspersor\_cont: std\_logic\_vector(3 downto 0);

signal selector\_switches: std\_logic\_vector (3 downto 0); -- sw(3 downto 0)

signal grados: integer range 0 to 180;

signal cont\_flancos: integer range 0 to 2000000; -- (20 ms) -> (50 Hz)

signal pwm\_longitud\_pulso: integer range 0 to 2000000; -- (20 ms) -> (50 Hz)

-- signals servomotor-reloj

signal segundos\_offset: std\_logic\_vector(3 downto 0); -- sw(15 downto 12)

signal suma\_o\_resta: std\_logic := '0';

signal cont\_base: integer range 0 to 400000000; -- lleva la cuenta del reloj, puesto para 1-4 seg...

signal tope\_freq: integer range 0 to 400000000;

-- signals del pulsador

signal estado\_pulsador: std\_logic\_vector (2 downto 0);

signal cont\_filtro: integer range 0 to 100000000;

signal salida: std\_logic;

signal flag\_suma: std\_logic;

signal flag\_resta: std\_logic;

signal freq\_min: integer range 0 to 100000000;

signal contador\_centenas: std\_logic\_vector (3 downto 0);

signal contador\_decenas: std\_logic\_vector (3 downto 0);

signal contador\_base\_enable: integer range 0 to 100000;

signal enable\_seg\_aux: std\_logic\_vector (3 downto 0);

signal dato: std\_logic\_vector (3 downto 0);

-- signals pulsador-pwm

signal contador\_decenas\_integer: integer range 0 to 9;

signal contador\_centenas\_integer: integer range 0 to 9;

signal numero\_int: integer range 0 to 200;

begin

-- ####################################################################

-- ####################################################################

-- LOGICA DEL SERVOMOTOR

-- ####################################################################

-- ####################################################################

segundos\_offset <= sw(15 downto 12);

selector\_aspersor\_mode <= sw(11);

selector\_input\_mode <= sw(10);

selector\_switches <= sw(3 downto 0);

-- process de designacià¸£à¸“n de grados por tiempo/switches/dedo

process(selector\_aspersor\_mode, aspersor\_cont, selector\_input\_mode, selector\_switches, suma\_o\_resta, numero\_int)

begin

-- grados por aspersor

if selector\_aspersor\_mode = '1' then

case aspersor\_cont is

when "0000" => grados <= 10;

when "0001" => grados <= 20;

when "0010" => grados <= 30;

when "0011" => grados <= 40;

when "0100" => grados <= 50;

when "0101" => grados <= 60;

when "0110" => grados <= 70;

when "0111" => grados <= 80;

when "1000" => grados <= 90;

when "1001" => grados <= 100;

when "1010" => grados <= 110;

when "1011" => grados <= 120;

when "1100" => grados <= 130;

when "1101" => grados <= 140;

when "1110" => grados <= 150;

when "1111" => grados <= 170;

when others => grados <= 10;

end case;

led(3 downto 0) <= aspersor\_cont;

led(8) <= suma\_o\_resta;

-- grados por switches

elsif selector\_input\_mode = '0' then

case selector\_switches is

when "0000" => grados <= 10;

when "0001" => grados <= 20;

when "0010" => grados <= 30;

when "0011" => grados <= 40;

when "0100" => grados <= 50;

when "0101" => grados <= 60;

when "0110" => grados <= 70;

when "0111" => grados <= 80;

when "1000" => grados <= 90;

when "1001" => grados <= 100;

when "1010" => grados <= 110;

when "1011" => grados <= 120;

when "1100" => grados <= 130;

when "1101" => grados <= 140;

when "1110" => grados <= 150;

when "1111" => grados <= 170;

when others => grados <= 10;

end case;

led(3 downto 0) <= "0000";

led(8) <= '0';

-- grados por dedo

else

grados <= numero\_int;

led(3 downto 0) <= "0000";

led(8) <= '0';

end if;

end process;

pwm\_longitud\_pulso <= grados \* 1111 + 50000;

-- process del automata del pwm del servo

process(clk, btnC)

begin

if btnC = '1' then

estado\_servo <= "00";

cont\_flancos <= 0;

elsif rising\_edge(clk) then

case estado\_servo is

when "00" =>

cont\_flancos <= 0;

estado\_servo <= "01";

when "01" =>

cont\_flancos <= 1;

estado\_servo <= "10";

when "10" =>

cont\_flancos <= cont\_flancos + 1;

if cont\_flancos = pwm\_longitud\_pulso then

estado\_servo <= "11";

else

estado\_servo <= "10";

end if;

when "11" =>

cont\_flancos <= cont\_flancos + 1;

if cont\_flancos = 2000000 then

estado\_servo <= "01";

else

estado\_servo <= "11";

end if;

when others =>

cont\_flancos <= 0;

estado\_servo <= "00";

end case;

end if;

end process;

-- process de salidas del servo

process(estado\_servo)

begin

case estado\_servo is

when "00" => servo <= '0';

when "01" => servo <= '1';

when "10" => servo <= '1';

when "11" => servo <= '0';

when others => servo <= '0';

end case;

end process;

-- ####################################################################

-- ####################################################################

-- LOGICA DEL RELOJ

-- ####################################################################

-- ####################################################################

-- proceso de reloj

process(btnC, clk)

begin

if btnC = '1' then

cont\_base <= 0;

elsif rising\_edge(clk) then

if cont\_base = tope\_freq then

cont\_base <= 0;

else

cont\_base <= cont\_base + 1;

end if;

end if;

end process;

-- process de cambio de vel.

process(segundos\_offset)

begin

if segundos\_offset = "XXX1" then

led(15 downto 12) <= "0001";

tope\_freq <= 100000000;

elsif segundos\_offset = "XX10" then

led(15 downto 12) <= "0010";

tope\_freq <= 200000000;

elsif segundos\_offset = "X100" then

led(15 downto 12) <= "0100";

tope\_freq <= 300000000;

elsif segundos\_offset = "1000" then

led(15 downto 12) <= "1000";

tope\_freq <= 400000000;

else

led(15 downto 12) <= "0000";

tope\_freq <= 100000000;

end if;

end process;

-- process de cambio de aspersor\_cont

process(btnC, clk)

begin

if btnC = '1' then

suma\_o\_resta <= '0';

aspersor\_cont <= "0000";

elsif rising\_edge(clk) then

if selector\_aspersor\_mode = '1' then

if cont\_base = tope\_freq then

if aspersor\_cont = "1111" then

suma\_o\_resta <= '1';

elsif aspersor\_cont = "0000" then

suma\_o\_resta <= '0';

end if;

if suma\_o\_resta = '0' and aspersor\_cont /= "1111" then

aspersor\_cont <= aspersor\_cont + 1;

elsif suma\_o\_resta = '1' and aspersor\_cont /= "0000" then

aspersor\_cont <= aspersor\_cont - 1;

end if;

end if;

else

suma\_o\_resta <= '0';

aspersor\_cont <= "0000";

end if;

end if;

end process;

-- ####################################################################

-- ####################################################################

-- LOGICA DEL PULSADOR

-- ####################################################################

-- ####################################################################

-- process del automata del pulsador

process(btnC, clk)

begin

if btnC = '1' then

estado\_pulsador <= "000";

cont\_filtro <= 0;

elsif rising\_edge(clk) then

case estado\_pulsador is

when "000" => -- INICIO

cont\_filtro <= 0;

if btnU = '1' or btnD = '1' then

estado\_pulsador <= "001";

else

estado\_pulsador <= "000";

end if;

when "001" => -- FILTRADO

cont\_filtro <= cont\_filtro + 1;

if (btnU = '1' or btnD = '1') and cont\_filtro < 100000 then

estado\_pulsador <= "001";

elsif (btnU = '1' or btnD = '1') and cont\_filtro = 100000 then

if btnU = '1'then

estado\_pulsador <= "010";

elsif btnD = '1' then

estado\_pulsador <= "100";

end if;

else

estado\_pulsador <= "000";

end if;

when "010" => -- UNO +

cont\_filtro <= 0;

if btnU = '1' then

estado\_pulsador <= "010";

else

estado\_pulsador <= "011";

end if;

when "011" => -- SUMA

cont\_filtro <= 0;

if btnU = '1' then

estado\_pulsador <= "001";

else

estado\_pulsador <= "000";

end if;

when "100" => -- UNO -

cont\_filtro <= 0;

if btnD = '1' then

estado\_pulsador <= "100";

else

estado\_pulsador <= "101";

end if;

when "101" => -- RESTA

cont\_filtro <= 0;

if btnD = '1' then

estado\_pulsador <= "001";

else

estado\_pulsador <= "000";

end if;

when others =>

cont\_filtro <= 0;

estado\_pulsador <= "000";

end case;

end if;

end process;

-- process de las salidas del pulsador

process(estado\_pulsador)

begin

case estado\_pulsador is

when "000" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "001" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "010" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "011" =>

salida <= '1';

flag\_suma <= '1';

flag\_resta <= '0';

when "100" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "101" =>

salida <= '1';

flag\_suma <= '0';

flag\_resta <= '1';

when others => salida <= '0';

end case;

end process;

-- process de sumar/restar decenas

process(btnC, clk)

begin

if btnC = '1' then

contador\_decenas <= "0001";

elsif rising\_edge(clk) then

if salida = '1' then

if flag\_suma = '1' then

if contador\_decenas = 7 and contador\_centenas = 1 then

contador\_decenas <= "0111";

elsif contador\_decenas = 9 then

contador\_decenas <= "0000";

else

contador\_decenas <= contador\_decenas + 1;

end if;

elsif flag\_resta = '1' then

if contador\_decenas = 1 and contador\_centenas = 0 then

contador\_decenas <= "0001";

elsif contador\_decenas = 0 then

contador\_decenas <= "1001";

else

contador\_decenas <= contador\_decenas - 1;

end if;

end if;

end if;

end if;

end process;

-- process de sumar/restar centenas

process(btnC, clk)

begin

if btnC = '1' then

contador\_centenas <= "0000";

elsif rising\_edge(clk) then

if salida = '1' then

if flag\_suma = '1' then

if contador\_decenas = 9 then

contador\_centenas <= contador\_centenas + 1;

end if;

elsif flag\_resta = '1' then

if contador\_centenas = 1 and contador\_decenas = 0 then

contador\_centenas <= contador\_centenas - 1;

end if;

end if;

end if;

end if;

end process;

contador\_decenas\_integer <= conv\_integer(contador\_decenas);

contador\_centenas\_integer <= conv\_integer(contador\_centenas);

numero\_int <= ((contador\_centenas\_integer \* 10) + contador\_decenas\_integer) \* 10;

-- proceso de frecuencia para el control del enable\_seg\_aux

process(clk, btnC)

begin

if btnC = '1' then

contador\_base\_enable <= 0;

elsif rising\_edge(clk) then

if contador\_base\_enable = 100000 then

contador\_base\_enable <= 0;

else

contador\_base\_enable <= contador\_base\_enable + 1;

end if;

end if;

end process;

-- proceso de control del enable\_seg\_aux

process(clk, btnC)

begin

if btnC = '1' then

enable\_seg\_aux <= "0111";

elsif rising\_edge(clk) then

if contador\_base\_enable = 100000 then

enable\_seg\_aux <= enable\_seg\_aux(2 downto 0) & enable\_seg\_aux(3);

end if;

end if;

end process;

an <= enable\_seg\_aux;

-- proceso de display de diferentes valores en diferentes siete\_segs

process(enable\_seg\_aux, contador\_decenas, contador\_centenas)

begin

if grados < 100 then

case enable\_seg\_aux is

when "0111" => dato <= "1111";

when "1011" => dato <= std\_logic\_vector(to\_unsigned(grados / 100, 4));

when "1101" => dato <= std\_logic\_vector(to\_unsigned(grados / 10, 4));

when "1110" => dato <= "0000";

when others => dato <= "1111";

end case;

else

case enable\_seg\_aux is

when "0111" => dato <= "1111";

when "1011" => dato <= std\_logic\_vector(to\_unsigned(grados / 100, 4));

when "1101" => dato <= std\_logic\_vector(to\_unsigned((grados / 10) - 10, 4));

when "1110" => dato <= "0000";

when others => dato <= "1111";

end case;

end if;

end process;

-- proceso de display de diferentes valores en diferentes siete\_segs

process(dato)

begin

case dato is

when "0000" => seg <= "0000001";

when "0001" => seg <= "1001111";

when "0010" => seg <= "0010010";

when "0011" => seg <= "0000110";

when "0100" => seg <= "1001100";

when "0101" => seg <= "0100100";

when "0110" => seg <= "1100000";

when "0111" => seg <= "0001111";

when "1000" => seg <= "0000000";

when "1001" => seg <= "0001100";

when others => seg <= "1111111";

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**dcmotor.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

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 instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity main is

port (

clk : in std\_logic;

sw : in STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- interruptores

btnU : in STD\_LOGIC; -- boton arriba

btnD : in STD\_LOGIC; -- boton abajo

btnL : in STD\_LOGIC; -- boton izquierda

btnR : in STD\_LOGIC; -- boton derecha

btnC : in STD\_LOGIC; -- boton central

led : out STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- leds

seg : out STD\_LOGIC\_VECTOR(6 DOWNTO 0); -- siete segmentos

dp : out STD\_LOGIC; -- punto decimal del seite segmentos

an : out STD\_LOGIC\_VECTOR(3 DOWNTO 0); -- control de 7-seg

dcmotor : out std\_logic\_vector (1 downto 0)

);

end main;

architecture Behavioral of main is

-- signals del pulsador de dedo

signal switch\_unidades\_o\_decenas: std\_logic; -- sw(0)

signal estado\_pulsador: std\_logic\_vector (2 downto 0);

signal cont\_filtro: integer range 0 to 100000000;

signal salida: std\_logic;

signal flag\_suma: std\_logic;

signal flag\_resta: std\_logic;

signal freq\_min: integer range 0 to 100000000;

signal contador\_unidades: std\_logic\_vector (3 downto 0);

signal contador\_decenas: std\_logic\_vector (3 downto 0);

signal contador\_base\_enable: integer range 0 to 100000;

signal enable\_seg\_aux: std\_logic\_vector (3 downto 0);

signal dato: std\_logic\_vector (3 downto 0);

-- signals pulsador-pwm

signal contador\_unidades\_integer: integer range 0 to 9;

signal contador\_decenas\_integer: integer range 0 to 9;

signal numero\_int: integer range 0 to 99;

-- signals del pwm

signal estado\_pwm: std\_logic\_vector (2 downto 0);

signal duty\_cycle: integer range 0 to 100; -- numero\_int

signal sentido\_giro: std\_logic; -- sw(1)

signal cont\_flancos: integer range 0 to 100000000;

signal pwm\_longitud\_pulso: integer range 0 to 100000000; -- longitud del pulso

signal pwm\_longitud\_ciclo: integer range 0 to 100000000; -- longitud del ciclo

signal pwm\_hz: integer range 0 to 500;

signal pwm\_out: std\_logic;

begin

-- ####################################################################

-- ####################################################################

-- LOGICA DEL PULSADOR

-- ####################################################################

-- ####################################################################

-- process del automata del pulsador

process(btnC, clk)

begin

if btnC = '1' then

estado\_pulsador <= "000";

cont\_filtro <= 0;

elsif rising\_edge(clk) then

case estado\_pulsador is

when "000" => -- INICIO

cont\_filtro <= 0;

if btnU = '1' or btnD = '1' then

estado\_pulsador <= "001";

else

estado\_pulsador <= "000";

end if;

when "001" => -- FILTRADO

cont\_filtro <= cont\_filtro + 1;

if (btnU = '1' or btnD = '1') and cont\_filtro < 100000 then

estado\_pulsador <= "001";

elsif (btnU = '1' or btnD = '1') and cont\_filtro = 100000 then

if btnU = '1'then

estado\_pulsador <= "010";

elsif btnD = '1' then

estado\_pulsador <= "100";

end if;

else

estado\_pulsador <= "000";

end if;

when "010" => -- UNO +

cont\_filtro <= 0;

if btnU = '1' then

estado\_pulsador <= "010";

else

estado\_pulsador <= "011";

end if;

when "011" => -- SUMA

cont\_filtro <= 0;

if btnU = '1' then

estado\_pulsador <= "001";

else

estado\_pulsador <= "000";

end if;

when "100" => -- UNO -

cont\_filtro <= 0;

if btnD = '1' then

estado\_pulsador <= "100";

else

estado\_pulsador <= "101";

end if;

when "101" => -- RESTA

cont\_filtro <= 0;

if btnD = '1' then

estado\_pulsador <= "001";

else

estado\_pulsador <= "000";

end if;

when others =>

cont\_filtro <= 0;

estado\_pulsador <= "000";

end case;

end if;

end process;

-- process de las salidas del pulsador

process(estado\_pulsador)

begin

case estado\_pulsador is

when "000" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "001" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "010" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "011" =>

salida <= '1';

flag\_suma <= '1';

flag\_resta <= '0';

when "100" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "101" =>

salida <= '1';

flag\_suma <= '0';

flag\_resta <= '1';

when others => salida <= '0';

end case;

end process;

switch\_unidades\_o\_decenas <= sw(0);

-- process de sumar/restar unidades (no da tiempo a llegar hasta 99 con weblab)

process(btnC, clk)

begin

if btnC = '1' then

contador\_unidades <= "0000";

elsif rising\_edge(clk) then

if salida = '1' and switch\_unidades\_o\_decenas = '0' then

if flag\_suma = '1' then

if contador\_unidades = 9 then

contador\_unidades <= "1001";

else

contador\_unidades <= contador\_unidades + 1;

end if;

elsif flag\_resta = '1' then

if contador\_unidades = 0 then

contador\_unidades <= "0000";

else

contador\_unidades <= contador\_unidades - 1;

end if;

end if;

end if;

end if;

end process;

-- process de sumar/restar decenas (no da tiempo a llegar hasta 99 con weblab)

process(btnC, clk)

begin

if btnC = '1' then

contador\_decenas <= "0000";

elsif rising\_edge(clk) then

if salida = '1' and switch\_unidades\_o\_decenas = '1' then

if flag\_suma = '1' then

if contador\_decenas = 9 then

contador\_decenas <= "1001";

else

contador\_decenas <= contador\_decenas + 1;

end if;

elsif flag\_resta = '1' then

if contador\_decenas = 0 then

contador\_decenas <= "0000";

else

contador\_decenas <= contador\_decenas - 1;

end if;

end if;

end if;

end if;

end process;

contador\_unidades\_integer <= conv\_integer(contador\_unidades);

contador\_decenas\_integer <= conv\_integer(contador\_decenas);

numero\_int <= (contador\_decenas\_integer \* 10) + contador\_unidades\_integer;

led(6 downto 0) <= std\_logic\_vector(to\_unsigned(numero\_int, 7));

-- proceso de frecuencia para el control del enable\_seg\_aux

process(clk, btnC)

begin

if btnC = '1' then

contador\_base\_enable <= 0;

elsif rising\_edge(clk) then

if contador\_base\_enable = 100000 then

contador\_base\_enable <= 0;

else

contador\_base\_enable <= contador\_base\_enable + 1;

end if;

end if;

end process;

-- proceso de control del enable\_seg\_aux

process(clk, btnC)

begin

if btnC = '1' then

enable\_seg\_aux <= "0111";

elsif rising\_edge(clk) then

if contador\_base\_enable = 100000 then

enable\_seg\_aux <= enable\_seg\_aux(2 downto 0) & enable\_seg\_aux(3);

end if;

end if;

end process;

an <= enable\_seg\_aux;

-- proceso de display de diferentes valores en diferentes siete\_segs

process(enable\_seg\_aux, contador\_unidades, contador\_decenas)

begin

case enable\_seg\_aux is

when "0111" => dato <= "1111";

when "1011" => dato <= "1111";

when "1101" => dato <= contador\_decenas;

when "1110" => dato <= contador\_unidades;

when others => dato <= "1111";

end case;

end process;

-- proceso de display de diferentes valores en diferentes siete\_segs

process(dato)

begin

case dato is

when "0000" => seg <= "0000001";

when "0001" => seg <= "1001111";

when "0010" => seg <= "0010010";

when "0011" => seg <= "0000110";

when "0100" => seg <= "1001100";

when "0101" => seg <= "0100100";

when "0110" => seg <= "1100000";

when "0111" => seg <= "0001111";

when "1000" => seg <= "0000000";

when "1001" => seg <= "0001100";

when others => seg <= "1111111";

end case;

end process;

-- ####################################################################

-- ####################################################################

-- LOGICA DEL MOTOR DC

-- ####################################################################

-- ####################################################################

duty\_cycle <= numero\_int;

sentido\_giro <= sw(1);

-- process del sentido de giro del motor

process(sentido\_giro)

begin

if sentido\_giro = '0' then

dcmotor <= "0" & pwm\_out;

else

dcmotor <= pwm\_out & "0";

end if;

end process;

pwm\_hz <= 200; -- (200 Hz) -> (500000 flancos) -> (0.5 ms)

-- process de definicion de pwm\_longitud\_ciclo como (frecuencia base de

-- la fpga) / (pwm\_hz)

process(pwm\_hz)

begin

pwm\_longitud\_ciclo <= 100000000 / pwm\_hz;

end process;

-- process de definicion de pwm\_longitud\_pulso como porcentaje X de

-- pwm\_longitud\_ciclo en funciรณn del valor de duty\_cycle

process(pwm\_longitud\_ciclo, duty\_cycle)

begin

pwm\_longitud\_pulso <= pwm\_longitud\_ciclo \* duty\_cycle / 100;

end process;

-- process del automata del pwm

process(clk, btnC)

begin

if btnC = '1' then

estado\_pwm <= "000";

cont\_flancos <= 0;

elsif rising\_edge(clk) then

case estado\_pwm is

when "000" =>

cont\_flancos <= 0;

if duty\_cycle /= 0 then

estado\_pwm <= "001";

else

estado\_pwm <= "100";

end if;

when "001" =>

cont\_flancos <= 1;

estado\_pwm <= "010";

when "010" =>

cont\_flancos <= cont\_flancos + 1;

if cont\_flancos < pwm\_longitud\_pulso then

estado\_pwm <= "010";

else

if duty\_cycle /= 100 then

estado\_pwm <= "011";

else

estado\_pwm <= "001";

end if;

end if;

when "011" =>

cont\_flancos <= cont\_flancos + 1;

if cont\_flancos < pwm\_longitud\_ciclo then

estado\_pwm <= "011";

else

if duty\_cycle /= 0 then

estado\_pwm <= "001";

else

estado\_pwm <= "100";

end if;

end if;

when "100" =>

cont\_flancos <= 1;

estado\_pwm <= "011";

when others =>

cont\_flancos <= 0;

estado\_pwm <= "000";

end case;

end if;

end process;

--process de las salidas del pwm

process(estado\_pwm)

begin

case estado\_pwm is

when "000" => pwm\_out <= '0';

when "001" => pwm\_out <= '1';

when "010" => pwm\_out <= '1';

when "011" => pwm\_out <= '0';

when "100" => pwm\_out <= '0';

when others => pwm\_out <= '0';

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**contador-dedo.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

entity project\_9 is

port(

clk: in std\_logic;

inicio: in std\_logic;

contador\_up: in std\_logic;

contador\_down: in std\_logic;

freq\_switch: in std\_logic;

seven\_seg: out std\_logic\_vector (6 downto 0);

enable\_seg: out std\_logic\_vector (3 downto 0)

);

end project\_9;

architecture Behavioral of project\_9 is

signal estado: std\_logic\_vector (2 downto 0);

signal cont\_filtro: integer range 0 to 100000000;

signal salida: std\_logic;

signal flag\_suma: std\_logic;

signal flag\_resta: std\_logic;

signal freq\_min: integer range 0 to 100000000;

signal contador\_unidades\_seg: std\_logic\_vector (3 downto 0);

signal contador\_decenas\_seg: std\_logic\_vector (3 downto 0);

signal contador\_base\_enable: integer range 0 to 100000;

signal enable\_seg\_aux: std\_logic\_vector (3 downto 0);

signal dato: std\_logic\_vector (3 downto 0);

begin

-- process del automata

process(inicio, clk)

begin

if inicio = '1' then

estado <= "000";

cont\_filtro <= 0;

elsif rising\_edge(clk) then

case estado is

when "000" => -- INICIO

cont\_filtro <= 0;

if contador\_up = '1' or contador\_down = '1' then

estado <= "001";

else

estado <= "000";

end if;

when "001" => -- FILTRADO

cont\_filtro <= cont\_filtro + 1;

if (contador\_up = '1' or contador\_down = '1') and cont\_filtro < freq\_min then

estado <= "001";

elsif (contador\_up = '1' or contador\_down = '1') and cont\_filtro = freq\_min then

if contador\_up = '1'then

estado <= "010";

elsif contador\_down = '1' then

estado <= "100";

end if;

else

estado <= "000";

end if;

when "010" => -- UNO +

cont\_filtro <= 0;

if contador\_up = '1' then

estado <= "010";

else

estado <= "011";

end if;

when "011" => -- SUMA

cont\_filtro <= 0;

if contador\_up = '1' then

estado <= "001";

else

estado <= "000";

end if;

when "100" => -- UNO -

cont\_filtro <= 0;

if contador\_down = '1' then

estado <= "100";

else

estado <= "101";

end if;

when "101" => -- RESTA

cont\_filtro <= 0;

if contador\_down = '1' then

estado <= "001";

else

estado <= "000";

end if;

when others =>

cont\_filtro <= 0;

estado <= "000";

end case;

end if;

end process;

-- process de las salidas

process(estado)

begin

case estado is

when "000" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "001" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "010" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "011" =>

salida <= '1';

flag\_suma <= '1';

flag\_resta <= '0';

when "100" =>

salida <= '0';

flag\_suma <= '0';

flag\_resta <= '0';

when "101" =>

salida <= '1';

flag\_suma <= '0';

flag\_resta <= '1';

when others => salida <= '0';

end case;

end process;

-- process de cambio de frecuencia minima para sumar/restar 1

process(freq\_switch)

begin

if freq\_switch = '0' then

freq\_min <= 100000;

else

freq\_min <= 100000000;

end if;

end process;

process(inicio, clk)

begin

if inicio='1' then

contador\_unidades\_seg <= "0000";

elsif rising\_edge(clk) then

if salida = '1' then

if flag\_suma = '1' then

if contador\_unidades\_seg = 9 and contador\_decenas\_seg < 9 then

contador\_unidades\_seg <= "0000";

elsif contador\_decenas\_seg = 9 and contador\_unidades\_seg = 9 then

contador\_unidades\_seg <= "1001";

else

contador\_unidades\_seg <= contador\_unidades\_seg + 1;

end if;

elsif flag\_resta = '1' then

if contador\_unidades\_seg = 0 and contador\_decenas\_seg > 0 then

contador\_unidades\_seg <= "1001";

elsif contador\_decenas\_seg = 0 and contador\_unidades\_seg = 0 then

contador\_unidades\_seg <= "0000";

else

contador\_unidades\_seg <= contador\_unidades\_seg - 1;

end if;

end if;

end if;

end if;

end process;

process(inicio, clk)

begin

if inicio = '1' then

contador\_decenas\_seg <= "0000";

elsif rising\_edge(clk) then

if salida = '1' then

if flag\_suma = '1' and contador\_unidades\_seg = 9 then

if contador\_decenas\_seg = 9 then

contador\_decenas\_seg <= "1001";

else

contador\_decenas\_seg <= contador\_decenas\_seg + 1;

end if;

elsif flag\_resta = '1' and contador\_unidades\_seg = 0 then

if contador\_decenas\_seg = 0 then

contador\_decenas\_seg <= "0000";

else

contador\_decenas\_seg <= contador\_decenas\_seg - 1;

end if;

end if;

end if;

end if;

end process;

process(clk, inicio)

begin

if inicio='1' then

contador\_base\_enable<=0;

elsif rising\_edge(clk) then

if contador\_base\_enable=100000 then

contador\_base\_enable<=0;

else

contador\_base\_enable<=contador\_base\_enable+1;

end if;

end if;

end process;

process(clk, inicio)

begin

if inicio='1' then

enable\_seg\_aux<="0111";

elsif rising\_edge(clk) then

if contador\_base\_enable=100000 then

enable\_seg\_aux<=enable\_seg\_aux(2 downto 0)&enable\_seg\_aux(3);

end if;

end if;

end process;

enable\_seg <= enable\_seg\_aux;

process(enable\_seg\_aux, contador\_unidades\_seg, contador\_decenas\_seg)

begin

case enable\_seg\_aux is

when "0111" => dato<="1111";

when "1011" => dato<="1111";

when "1101" => dato<=contador\_decenas\_seg;

when "1110" => dato<=contador\_unidades\_seg;

when others => dato<="1111";

end case;

end process;

process(dato)

begin

case dato is

when "0000" => seven\_seg<="0000001";

when "0001" => seven\_seg<="1001111";

when "0010" => seven\_seg<="0010010";

when "0011" => seven\_seg<="0000110";

when "0100" => seven\_seg<="1001100";

when "0101" => seven\_seg<="0100100";

when "0110" => seven\_seg<="1100000";

when "0111" => seven\_seg<="0001111";

when "1000" => seven\_seg<="0000000";

when "1001" => seven\_seg<="0001100";

when others => seven\_seg<="1111111";

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**reloj.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

entity project\_8 is

port(

clk: in std\_logic;

init: in std\_logic;

select\_mode: in std\_logic;

fast: in std\_logic;

pausa: in std\_logic;

siete\_seg: out std\_logic\_vector (6 downto 0);

enable\_seg: out std\_logic\_vector (3 downto 0)

);

end project\_8;

architecture Behavioral of project\_8 is

signal cont\_centesimas: std\_logic\_vector (3 downto 0);

signal cont\_decimas: std\_logic\_vector (3 downto 0);

signal cont\_segs\_unidades: std\_logic\_vector (3 downto 0);

signal cont\_segs\_decenas: std\_logic\_vector (3 downto 0);

signal cont\_mins\_unidades: std\_logic\_vector (3 downto 0);

signal cont\_mins\_decenas: std\_logic\_vector (3 downto 0);

signal cont\_horas\_unidades: std\_logic\_vector (3 downto 0);

signal cont\_horas\_decenas: std\_logic\_vector (3 downto 0);

signal cont\_centesimas\_temp: std\_logic\_vector (3 downto 0);

signal cont\_decimas\_temp: std\_logic\_vector (3 downto 0);

signal cont\_segs\_unidades\_temp: std\_logic\_vector (3 downto 0);

signal cont\_segs\_decenas\_temp: std\_logic\_vector (3 downto 0);

signal cont\_mins\_unidades\_temp: std\_logic\_vector (3 downto 0);

signal cont\_mins\_decenas\_temp: std\_logic\_vector (3 downto 0);

signal cont\_horas\_unidades\_temp: std\_logic\_vector (3 downto 0);

signal cont\_horas\_decenas\_temp: std\_logic\_vector (3 downto 0);

-- signal cont\_base: integer range 0 to 100000000; -- lleva la cuenta del reloj, puesto para 1 seg...

signal cont\_base: integer range 0 to 1000000; -- lleva la cuenta del reloj, puesto para 1 centesima...

signal tope\_freq: integer range 0 to 1000000;

signal enable\_aux: std\_logic\_vector (3 downto 0);

signal cont\_base\_enable: integer range 0 to 100000; -- 100,000,000 / 100,000 = 1kHz de freq (cada siete\_seg encendido este tiempo / (num de siete\_segs)...)

signal sal\_mux: std\_logic\_vector (3 downto 0);

begin

-- proceso de reloj

process(init, clk)

begin

if init = '1' then

cont\_base <= 0;

elsif rising\_edge(clk) then

if cont\_base = tope\_freq then

cont\_base <= 0;

else

cont\_base <= cont\_base + 1;

end if;

end if;

end process;

-- proceso de cambio de vel.

process(fast)

begin

if fast = '0' then

tope\_freq <= 1000000;

else

tope\_freq <= 100;

end if;

end process;

process(pausa, select\_mode, clk)

begin

if pausa = '0' and rising\_edge(clk) then

if select\_mode = '0' then

cont\_centesimas\_temp <= cont\_centesimas;

cont\_decimas\_temp <= cont\_decimas;

cont\_segs\_unidades\_temp <= cont\_segs\_unidades;

cont\_segs\_decenas\_temp <= cont\_segs\_decenas;

else

cont\_mins\_unidades\_temp <= cont\_mins\_unidades;

cont\_mins\_decenas\_temp <= cont\_mins\_decenas;

cont\_horas\_unidades\_temp <= cont\_horas\_unidades;

cont\_horas\_decenas\_temp <= cont\_horas\_decenas;

end if;

else

end if;

end process;

-- proceso de monitoreo de segundos.unidades

process(init, clk)

begin

if init = '1' then

cont\_centesimas <= "0000";

elsif rising\_edge(clk) then

-- si ha pasado un segundo

if cont\_base = tope\_freq then

if cont\_centesimas = "1001" then

cont\_centesimas <= "0000";

else

cont\_centesimas <= cont\_centesimas + 1;

end if;

end if;

end if;

end process;

-- proceso de monitoreo de segundos.decenas

process(init, clk)

begin

if init = '1' then

cont\_decimas <= "0000";

elsif rising\_edge(clk) then

-- si ha pasado un segundo y las segundos.unidades = 9

if cont\_base = tope\_freq and cont\_centesimas = "1001" then

if cont\_decimas = "1001" then

cont\_decimas <= "0000";

else

cont\_decimas <= cont\_decimas + 1;

end if;

end if;

end if;

end process;

-- proceso de monitoreo de minutos.unidades

process(init, clk)

begin

if init = '1' then

cont\_segs\_unidades <= "0000";

elsif rising\_edge(clk) then

-- si ha pasado un segundo y las segundos.unidades = 9 y segundos.decenas = 5

if cont\_base = tope\_freq and cont\_centesimas = "1001" and cont\_decimas = "1001" then

if cont\_segs\_unidades = "1001" then

cont\_segs\_unidades <= "0000";

else

cont\_segs\_unidades <= cont\_segs\_unidades + 1;

end if;

end if;

end if;

end process;

-- proceso de monitoreo de segundos.decenas

process(init, clk)

begin

if init = '1' then

cont\_segs\_decenas <= "0000";

elsif rising\_edge(clk) then

-- si ha pasado un segundo y las segundos.unidades = 9 y segundos.decenas = 5 y minutos.unidades = 9

if cont\_base = tope\_freq and cont\_centesimas = "1001" and cont\_decimas = "1001" and cont\_segs\_unidades = "1001" then

if cont\_segs\_decenas = "0101" then

cont\_segs\_decenas <= "0000";

else

cont\_segs\_decenas <= cont\_segs\_decenas + 1;

end if;

end if;

end if;

end process;

-- proceso de monitoreo de minutos.unidades

process(init, clk)

begin

if init = '1' then

cont\_mins\_unidades <= "0000";

elsif rising\_edge(clk) then

-- si ha pasado un segundo y las segundos.unidades = 9 y segundos.decenas = 5 y minutos.unidades = 9

if cont\_base = tope\_freq and cont\_centesimas = "1001" and cont\_decimas = "1001" and cont\_segs\_unidades = "1001" and cont\_segs\_decenas = "0101" then

if cont\_mins\_unidades = "1001" then

cont\_mins\_unidades <= "0000";

else

cont\_mins\_unidades <= cont\_mins\_unidades + 1;

end if;

end if;

end if;

end process;

-- proceso de monitoreo de minutos.decenas

process(init, clk)

begin

if init = '1' then

cont\_mins\_decenas <= "0000";

elsif rising\_edge(clk) then

-- si ha pasado un segundo y las segundos.unidades = 9 y segundos.decenas = 5 y minutos.unidades = 9

if cont\_base = tope\_freq and cont\_centesimas = "1001" and cont\_decimas = "1001" and cont\_segs\_unidades = "1001" and cont\_segs\_decenas = "0101" and cont\_mins\_unidades = "1001" then

if cont\_mins\_decenas = "0101" then

cont\_mins\_decenas <= "0000";

else

cont\_mins\_decenas <= cont\_mins\_decenas + 1;

end if;

end if;

end if;

end process;

-- proceso de monitoreo de horas.unidades

process(init, clk)

begin

if init = '1' then

cont\_horas\_unidades <= "0000";

elsif rising\_edge(clk) then

-- si ha pasado un segundo y las segundos.unidades = 9 y segundos.decenas = 5 y minutos.unidades = 9

if cont\_base = tope\_freq and cont\_centesimas = "1001" and cont\_decimas = "1001" and cont\_segs\_unidades = "1001" and cont\_segs\_decenas = "0101" and cont\_mins\_unidades = "1001" and cont\_mins\_decenas = "0101" then

if cont\_horas\_unidades = "1001" then

cont\_horas\_unidades <= "0000";

elsif cont\_horas\_unidades = "0011" and cont\_horas\_decenas = "0010" then

cont\_horas\_unidades <= "0000";

else

cont\_horas\_unidades <= cont\_horas\_unidades + 1;

end if;

end if;

end if;

end process;

-- proceso de monitoreo de horas.decenas

process(init, clk)

begin

if init = '1' then

cont\_horas\_decenas <= "0000";

elsif rising\_edge(clk) then

-- si ha pasado un segundo y las segundos.unidades = 9 y segundos.decenas = 5 y minutos.unidades = 9

if cont\_base = tope\_freq and cont\_centesimas = "1001" and cont\_decimas = "1001" and cont\_segs\_unidades = "1001" and cont\_segs\_decenas = "0101" and cont\_mins\_unidades = "1001" and cont\_mins\_decenas = "0101" and (cont\_horas\_unidades = "1001" or cont\_horas\_unidades = "0011") then

if cont\_horas\_decenas = "0010" then

cont\_horas\_decenas <= "0000";

elsif cont\_horas\_unidades = "1001" then

cont\_horas\_decenas <= cont\_horas\_decenas + 1;

end if;

end if;

end if;

end process;

-- proceso de conteo y modificacion del cont\_base\_enable (el que da la freq de 1kHz para cambio de siete\_seg)

process(init, clk)

begin

if init = '1' then

cont\_base\_enable <= 0;

elsif rising\_edge(clk) then

if cont\_base\_enable = 1000000 then

cont\_base\_enable <= 0;

else

cont\_base\_enable <= cont\_base\_enable + 1;

end if;

end if;

end process;

-- modificacion de la signal para ir de siete\_seg en siete\_seg (rotacion izquierda 1000 veces por segundo)

process(init, clk)

begin

if init = '1' then

enable\_aux <= "1110"; -- TODOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO

elsif rising\_edge(clk) then

if cont\_base\_enable = 100000 then

enable\_aux <= enable\_aux(2 downto 0) & enable\_aux(3); -- desplazamiento a la izquierda

-- enable\_aux <= enable\_aux(0) & enable\_aux(3 downto 1); -- desplazamiento a la derecha

end if;

end if;

end process;

enable\_seg <= enable\_aux;

--multiplexado de las entradas al 7-seg

process(enable\_aux, pausa, select\_mode, cont\_centesimas, cont\_decimas, cont\_segs\_unidades, cont\_segs\_decenas, cont\_mins\_unidades, cont\_mins\_decenas, cont\_horas\_unidades, cont\_horas\_decenas)

begin

if pausa = '0' then

if select\_mode = '0' then

case enable\_aux is

when "1110" => sal\_mux <= cont\_mins\_unidades;

when "1101" => sal\_mux <= cont\_mins\_decenas;

when "1011" => sal\_mux <= cont\_horas\_unidades;

when "0111" => sal\_mux <= cont\_horas\_decenas;

when others => sal\_mux <= "0000";

end case;

else

case enable\_aux is

when "1110" => sal\_mux <= cont\_centesimas;

when "1101" => sal\_mux <= cont\_decimas;

when "1011" => sal\_mux <= cont\_segs\_unidades;

when "0111" => sal\_mux <= cont\_segs\_decenas;

when others => sal\_mux <= "0000";

end case;

end if;

else

if select\_mode = '0' then

case enable\_aux is

when "1110" => sal\_mux <= cont\_mins\_unidades\_temp;

when "1101" => sal\_mux <= cont\_mins\_decenas\_temp;

when "1011" => sal\_mux <= cont\_horas\_unidades\_temp;

when "0111" => sal\_mux <= cont\_horas\_decenas\_temp;

when others => sal\_mux <= "0000";

end case;

else

case enable\_aux is

when "1110" => sal\_mux <= cont\_centesimas\_temp;

when "1101" => sal\_mux <= cont\_decimas\_temp;

when "1011" => sal\_mux <= cont\_segs\_unidades\_temp;

when "0111" => sal\_mux <= cont\_segs\_decenas\_temp;

when others => sal\_mux <= "0000";

end case;

end if;

end if;

end process;

process(sal\_mux)

begin

case sal\_mux is

when "0000" => siete\_seg <= "0000001";

when "0001" => siete\_seg <= "1001111";

when "0010" => siete\_seg <= "0010010";

when "0011" => siete\_seg <= "0000110";

when "0100" => siete\_seg <= "1001100";

when "0101" => siete\_seg <= "0100100";

when "0110" => siete\_seg <= "1100000";

when "0111" => siete\_seg <= "0001111";

when "1000" => siete\_seg <= "0000000";

when "1001" => siete\_seg <= "0001100";

when others => siete\_seg <= "1111111";

end case;

end process;

end behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**decimalesABenBCDVerMayor.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

--use IEEE.std\_logic\_signed.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity project\_entrega2\_1 is

port(

num\_a: in std\_logic\_vector (3 downto 0);

num\_b: in std\_logic\_vector (3 downto 0);

siete\_seg: out std\_logic\_vector (6 downto 0);

enable\_seg: out std\_logic\_vector (3 downto 0)

);

end project\_entrega2\_1;

architecture Behavioral of project\_entrega2\_1 is

signal numero: std\_logic\_vector (3 downto 0);

begin

enable\_seg <= "1110";

process(num\_a, num\_b, numero)

begin

if num\_a > num\_b then

numero <= num\_a;

elsif num\_b > num\_a then

numero <= num\_b;

else

numero <= "1111";

end if;

end process;

process(numero)

begin

case numero is

when "0000" => siete\_seg <= "1000000";

when "0001" => siete\_seg <= "1111001";

when "0010" => siete\_seg <= "0100100";

when "0011" => siete\_seg <= "0110000";

when "0100" => siete\_seg <= "0011001";

when "0101" => siete\_seg <= "0010010";

when "0110" => siete\_seg <= "0000011";

when "0111" => siete\_seg <= "1111000";

when "1000" => siete\_seg <= "0000000";

when "1001" => siete\_seg <= "0011000";

when others => siete\_seg <= "1111111";

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**restadeNumsBCD.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

--use IEEE.std\_logic\_signed.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity project\_entrega2\_1 is

port(

num\_a: in std\_logic\_vector (3 downto 0);

num\_b: in std\_logic\_vector (3 downto 0);

siete\_seg: out std\_logic\_vector (6 downto 0);

enable\_seg: out std\_logic\_vector (3 downto 0)

);

end project\_entrega2\_1;

architecture Behavioral of project\_entrega2\_1 is

signal numero: std\_logic\_vector (3 downto 0);

begin

enable\_seg <= "1110";

process(num\_a, num\_b, numero)

begin

if num\_a >= num\_b then

numero <= num\_a - num\_b;

else

numero <= num\_b - num\_a;

end if;

end process;

process(numero)

begin

case numero is

when "0000" => siete\_seg <= "1000000";

when "0001" => siete\_seg <= "1111001";

when "0010" => siete\_seg <= "0100100";

when "0011" => siete\_seg <= "0110000";

when "0100" => siete\_seg <= "0011001";

when "0101" => siete\_seg <= "0010010";

when "0110" => siete\_seg <= "0000011";

when "0111" => siete\_seg <= "1111000";

when "1000" => siete\_seg <= "0000000";

when "1001" => siete\_seg <= "0011000";

when others => siete\_seg <= "1111111";

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**eldelostres.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

--use IEEE.std\_logic\_signed.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity project\_entrega2\_1 is

port(

num\_a: in std\_logic\_vector (3 downto 0);

num\_b: in std\_logic\_vector (3 downto 0);

num\_c: in std\_logic\_vector (3 downto 0);

siete\_seg: out std\_logic\_vector (6 downto 0);

enable\_seg: out std\_logic\_vector (3 downto 0)

);

end project\_entrega2\_1;

architecture Behavioral of project\_entrega2\_1 is

signal numero: std\_logic\_vector (3 downto 0);

begin

enable\_seg <= "1110";

process(num\_a, num\_b, numero)

begin

if num\_a >= num\_b then

if num\_a >= num\_c then

numero <= num\_a;

else

numero <= num\_c;

end if;

else

if num\_b >= num\_c then

numero <= num\_b;

else

numero <= num\_c;

end if;

end if;

end process;

process(numero)

begin

case numero is

when "0000" => siete\_seg <= "1000000";

when "0001" => siete\_seg <= "1111001";

when "0010" => siete\_seg <= "0100100";

when "0011" => siete\_seg <= "0110000";

when "0100" => siete\_seg <= "0011001";

when "0101" => siete\_seg <= "0010010";

when "0110" => siete\_seg <= "0000011";

when "0111" => siete\_seg <= "1111000";

when "1000" => siete\_seg <= "0000000";

when "1001" => siete\_seg <= "0011000";

when others => siete\_seg <= "1111111";

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**sumador-restador-signo.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

-- ZONA LIBRERIAS

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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

-- ZONA DECLARACION ENTRADAS Y SALIDAS

entity project\_1 is

port(

num\_a: in std\_logic\_vector (3 downto 0);

num\_b: in std\_logic\_vector (3 downto 0);

selector: in std\_logic;

siete\_seg: out std\_logic\_vector (6 downto 0);

enable\_seg: out std\_logic\_vector (3 downto 0);

--blank\_input: in std\_logic;

suma: out std\_logic\_vector (3 downto 0);

signo: out std\_logic

);

end project\_1;

architecture Behavioral of project\_1 is

-- ZONA SIGNALS

signal resultado: std\_logic\_vector (3 downto 0);

signal resultado\_entero: integer range -15 to 30;

signal resultado\_entero\_final: integer range -15 to 30;

signal num\_a\_entero: integer range 0 to 15;

signal num\_b\_entero: integer range 0 to 15;

signal blank\_input: std\_logic;

begin

-- ZONA DECLARACION VHDL

enable\_seg <= "1110";

num\_a\_entero <= to\_integer(unsigned(num\_a));

num\_b\_entero <= to\_integer(unsigned(num\_b));

process(num\_a\_entero, num\_b\_entero, selector)

begin

if selector='0' then

resultado\_entero <= num\_a\_entero + num\_b\_entero;

else

resultado\_entero <= num\_a\_entero - num\_b\_entero;

end if;

end process;

suma <= std\_logic\_vector(to\_signed(resultado\_entero, 4));

process(resultado\_entero)

begin

if resultado\_entero < 0 then

signo <= '1';

resultado\_entero\_final <= - resultado\_entero;

else

signo <= '0';

resultado\_entero\_final <= resultado\_entero;

end if;

end process;

process(resultado\_entero\_final)

begin

if resultado\_entero\_final <= 9 and resultado\_entero\_final >= -9 then

blank\_input <= '0';

else

blank\_input <= '1';

end if;

end process;

process(resultado\_entero\_final, blank\_input)

begin

if blank\_input = '0' then

case resultado\_entero\_final is

when 0 => siete\_seg <= "1000000";

when 1 => siete\_seg <= "1111001";

when 2 => siete\_seg <= "0100100";

when 3 => siete\_seg <= "0110000";

when 4 => siete\_seg <= "0011001";

when 5 => siete\_seg <= "0010010";

when 6 => siete\_seg <= "0000011";

when 7 => siete\_seg <= "1111000";

when 8 => siete\_seg <= "0000000";

when 9 => siete\_seg <= "0011000";

when others => siete\_seg <= "0000110";

end case;

else

siete\_seg <= "1111111";

end if;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**diamantes.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

--use IEEE.std\_logic\_signed.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity project\_2\_diamante is

port(

sensor\_bajo: in std\_logic;

sensor\_medio: in std\_logic;

sensor\_alto: in std\_logic;

sensor\_peso: in std\_logic;

salida\_bajo: out std\_logic;

salida\_medio: out std\_logic;

salida\_alto: out std\_logic;

salida\_rechazado: out std\_logic;

salida\_error: out std\_logic;

siete\_seg: out std\_logic\_vector (6 downto 0);

enable\_seg: out std\_logic\_vector (3 downto 0)

);

end project\_2\_diamante;

architecture Behavioral of project\_2\_diamante is

signal sensores: std\_logic\_vector (3 downto 0);

signal flags: std\_logic\_vector (4 downto 0); -- (0) baj (1) med (2) alt (3) rech (4) err

begin

enable\_seg <= "1110";

-- procesamos entradas

sensores <= sensor\_peso & sensor\_alto & sensor\_medio & sensor\_bajo;

process(sensores)

begin

case sensores is

-- bajo

when "0110" => flags <= "00001";

-- medio

when "1100" => flags <= "00010";

-- alto

when "1000" => flags <= "00100";

-- rechazado

when "1110" | "1111" | "0111" | "0000" | "0100" => flags <= "01000";

-- error

when others => flags <= "10000";

end case;

end process;

-- procesamos salidas

salida\_bajo <= flags(0);

salida\_medio <= flags(1);

salida\_alto <= flags(2);

salida\_rechazado <= flags(3);

salida\_error <= flags(4);

process(flags)

begin

case flags is

-- bajo

when "00001" => siete\_seg <= "0001100";

-- medio

when "00010" => siete\_seg <= "0001001";

-- alto

when "00100" => siete\_seg <= "0000010";

-- rechazado

when "01000" => siete\_seg <= "0101111";

-- error

when "10000" => siete\_seg <= "0000110";

when others => siete\_seg <= "1111111";

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**notas-alumnos.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

--use IEEE.std\_logic\_signed.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity project\_3\_notas is

port (

nota: in std\_logic\_vector (3 downto 0);

siete\_seg: out std\_logic\_vector (6 downto 0);

enable\_seg: out std\_logic\_vector (3 downto 0)

);

end project\_3\_notas;

architecture Behavioral of project\_3\_notas is

begin

enable\_seg <= "1110";

process(nota)

begin

case nota is

when "0000" => siete\_seg <= "0111111";

when "0001" => siete\_seg <= "0111111";

when "0010" => siete\_seg <= "0111111";

when "0011" => siete\_seg <= "0111111";

when "0100" => siete\_seg <= "0111111";

when "0101" => siete\_seg <= "1111110";

when "0110" => siete\_seg <= "1111100";

when "0111" => siete\_seg <= "1111000";

when "1000" => siete\_seg <= "1111000";

when "1001" => siete\_seg <= "1110000";

when "1010" => siete\_seg <= "1000000";

when others => siete\_seg <= "1111111";

end case;

end process;

end Behavioral;

**###########################################################################**

**###########################################################################**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**cafetera.vhd**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**###########################################################################**

**###########################################################################**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

--use IEEE.std\_logic\_signed.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 leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity project\_4\_cafetera is

port(

INT: in std\_logic; -- encendido/apagado

NB: in std\_logic; -- sensor agua bajo

NA: in std\_logic; -- sensor agua alto

C: in std\_logic; -- cantidad cafe

siete\_seg: out std\_logic\_vector (6 downto 0);

enable\_seg: out std\_logic\_vector (3 downto 0)

);

end project\_4\_cafetera;

architecture Behavioral of project\_4\_cafetera is

begin

enable\_seg <= "1110";

process(INT, NB, NA, C)

begin

-- cafetera apagada

if INT = '0' then

siete\_seg <= "1000000"; -- cero

-- cafetera encendida

else

-- no hay agua

if NA = '0' and NB = '0' then

siete\_seg <= "0000110"; -- "E" error

-- hay agua

else

-- cafe suave

if NA = '1' and NB = '1' and C = '0' then

siete\_seg <= "1110111"; -- segmento d

-- cafe normal

elsif (NA = '1' and NB = '1' and C = '1') or (NA = '0' and NB = '1' and C = '0') then

siete\_seg <= "0110111"; -- segmento d y g

-- cafe fuerte

elsif NA = '0' and NB = '1' and C = '1' then

siete\_seg <= "0110110"; -- segmento d y g y a

-- condiciones libres...

else

siete\_seg <= "0000110"; -- "E" error

end if;

end if;

end if;

end process;

end Behavioral;