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(13);

modo\_lento\_rapido <= sw(12);

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;