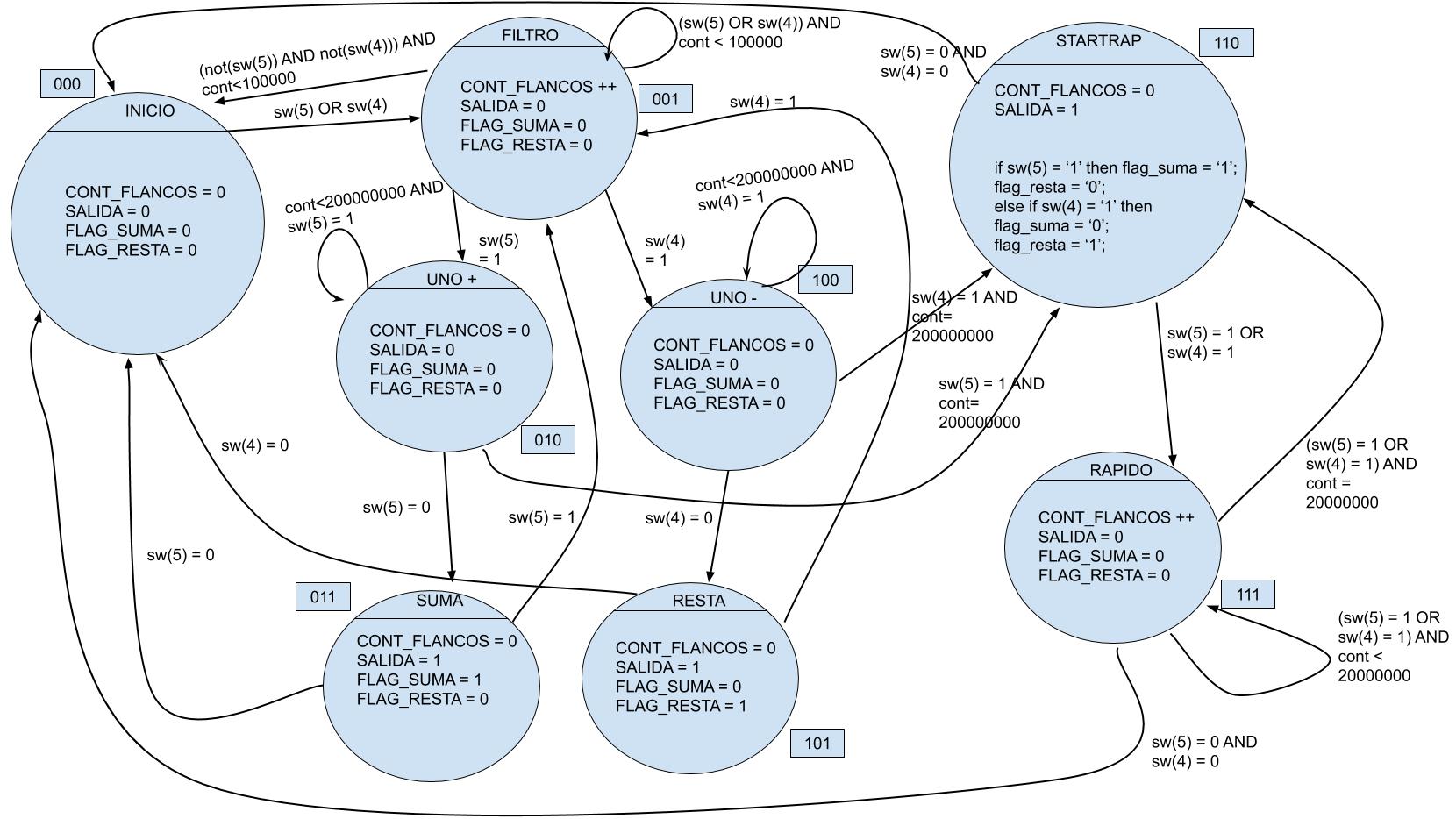
Cuarta sesión remota 2019-2020

Iratxe Campo Larrinaga

Asier Fernández de Lecea

* **Máquina de estados**

****

* **Diseño y prueba del nuevo autómata de detección de pulso para velocidad rápida.**

Hecho.

* **Modificación del autómata de clase de forma personal. OPCIONAL.**

Modificado para poder sumar o restar.

* **Integración del pulso de cuenta rápida en algún ejercicio anterior.**

Servomotor, junto con los otros dos métodos de cambio de posición del servomotor. En total, el servomotor contiene lo siguiente: modo aspersor, modo manual con los switches 3 downto 0 y modo suma/resta con los switches 5 y 4. Dado que los botones de weblab no detectan pulsaciones largas, hemos puesto estos switches 5 y 4 para poder detectar las “pulsaciones” de más de dos segundos.

**Fichero 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;