****

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

LABORATORIO SISTEMAS DIGITALES II

**PARALELO:**

Teórico #2, Practico 101

**PROFESOR:**

Alexander Prieto León

**ESTUDIANTES:**

Isaac Armijos Guevara

Joffre Prieto Chuquimarca

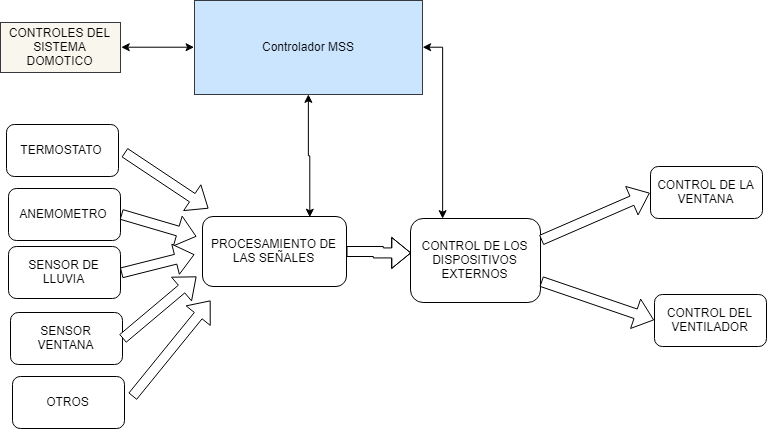
**TÍTULO DEL PROYECTO:**

CONTROL DOMÓTICO DE TOLDO

**FECHA ENTREGA:**

14 de Agosto de 2019

1. Diagrama de Bloques que incluya las variables de entrada, salida y señales internas del Sistema Digital.



1. Listado que contenga todas las señales de entrada, salida y señales internas (Lista y explicación de cada una).

**Controles del sistema domótico**: Contiene las señales de inicialización del sistema, así como variables de control.

**Termostato**: Señal digital que indica el valor umbral del calor del interior de la casa a 27 grados.

**Anemómetro**: Señal digital situado en el exterior de la casa con valor regulable, que mide la variación del viento.

**Sensor de lluvia**: Señal digital de lluvia situado en el exterior y a cuál se activa cuando llueve.

**Sensor de ventana**: Señal digital que se activa cuando la ventana se abre.

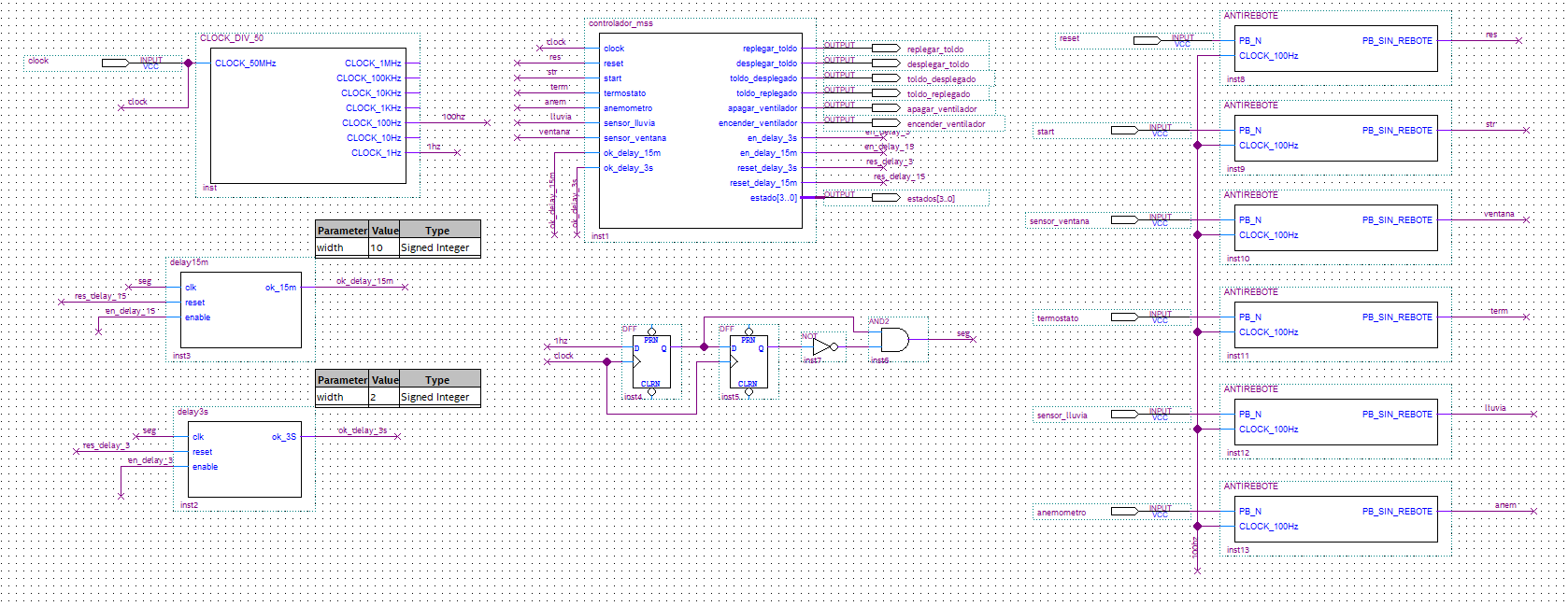
**Procesamiento de las señales**: Como son entradas directamente de dispositivos pasivos, es requerido un circuito anti-rebote para filtrar los pulsos generados físicamente en cada switch.

**Control de la ventana**: Controlar el motor de la apertura o cierre del toldo situado en la ventana.

**Control del ventilador**: Control externo directamente hacia el ventilador

**Estado[3..0]:** Señal que muestra en que estado se encuentra la maquina secuencial

1. Partición Funcional que incluya las variables de entrada, salida y señales internas del Sistema Digital.



1. Listado que contenga todas las señales de entrada, salida y señales internas de cada bloque (Lista y explicación de cada una).

**termostato**: Señal digital que indica el valor umbral del calor del interior de la casa a 27 grados.

**anemómetro**: Señal digital situado en el exterior de la casa con valor regulable, que mide la variación del viento.

**sensor\_lluvia**: Señal digital de lluvia situado en el exterior y a cuál se activa cuando llueve.

**Sensor-ventana**: Señal digital que se activa cuando la ventana se abre.

**clock**: Señal de clock para el controlador mss del circuito digital, que se encargará del procesamiento de las señales.

**resetn**: señal de reseteo, para la inicialización del controlador con el asm interno del sistema digital.

**start**: Señal para la inicialización del controlador mss.

**ok\_delay\_15m**: Señal de activación del delay configurado a 15 min, que sirven para el conteo de la puerta abierta y verificar si presenta algún error.

**ok\_delay\_3s**: Señal de activación del delay configurado a 3segundos, que sirven para simulador el tiempo en la cual se demora en abrirse o cerrarse el toldo.

**replegar\_toldo**: Señal que va directamente conectado al circuito de control para el motor que replegara el toldo.

**desplegar\_toldo**: Señal que va directamente conectado al circuito de control para el motor que desplegara el toldo.

**encender\_ventilador:** Señal digital directamente conectado para encender el ventilador encargado de extraer el aire caliente.

**apagar\_ventilador:** Señal digital directamente conectado para apagar el ventilador encargado de extraer el aire caliente.

**toldo\_replegado:** Señal digital que indica cuando el toldo esta replegado.

**toldo\_desplegado:** Señal digital que indica cual el toldo esta desplegado.

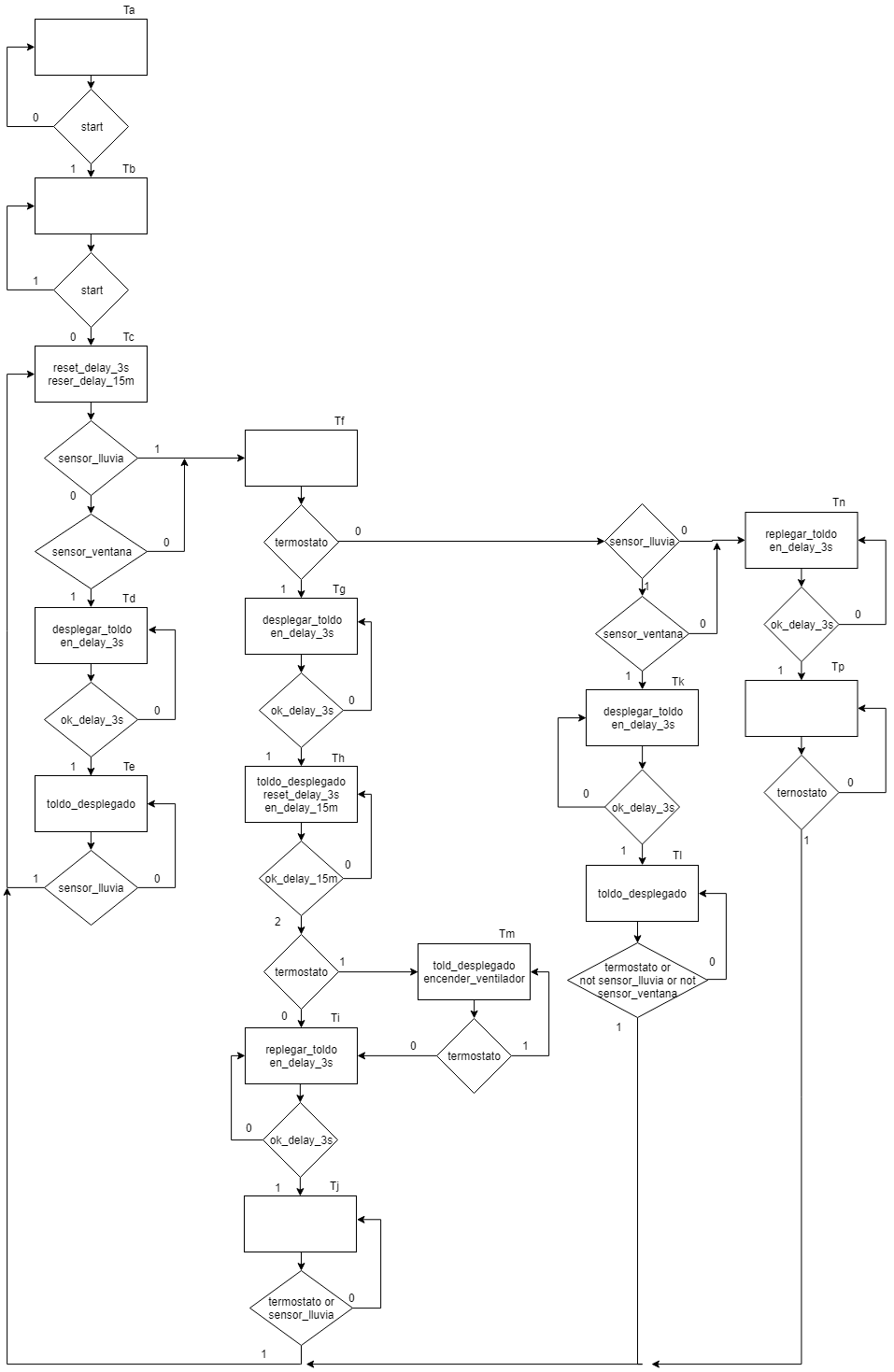
**en\_delay\_15m**: Señal digital para activar el contador del delay correspondiente a 15 min, que sirve para el conteo de la puerta abierta y verificar si presenta algún error.

**en\_delay\_3s**: Señal digital para activar el contador del delay correspondiente a 3 segundos, que sirven para simulador el tiempo en la cual se demora en abrirse o cerrarse el toldo.

**reset\_delay\_15m**: Señal digital para resetear el contador del delay correspondiente a 15 min.

**reset\_delay\_3s**: Señal digital para resetear el contador del delay correspondiente a 3 segundos.

1. Diagrama ASM del Controlador en Visio o cualquier otra herramienta utilizada para elaborar diagramas de flujo, indicando claramente todas las señales de entrada y salida utilizadas (con su respectiva explicación a detalle).



**Start**: Señal para la inicialización del controlador mss con antirrebote configurado.

**Termostato**: Señal digital de entrada al mss que indica el valor umbral del calor del interior de la casa a 27 grados.

**Sensor de lluvia**: Señal digital de entrada al mss de lluvia situado en el exterior y a cuál se activa cuando llueve.

**Sensor de ventana**: Señal digital de entrada al mss que se activa cuando la ventana se abre.

**Ok\_delay\_15m**: Señal de entrada al mss de activación del delay configurado a 15 min.

**Ok\_delay\_3s**: Señal de entrada al mss de activación del delay configurado a 3segundos.

**Replegar\_toldo**: Señal de salida del mss que va directamente conectado al circuito de control para el motor que replegara el toldo.

**Desplegar\_toldo**: Señal de salida del mss que va directamente conectado al circuito de control para el motor que desplegara el toldo.

**Encender ventilador:** Señal de salida del mss digital directamente conectado para encender el ventilador encargado de extraer el aire caliente.

**Apagar ventilador:** Señal de salida del mss digital directamente conectado para apagar el ventilador encargado de extraer el aire caliente.

**En\_delay\_15m**: Señal digital de salida del mss para activar el contador del delay correspondiente a 15 min.

**En\_delay\_3s**: Señal digital de salida del mss para activar el contador del delay correspondiente a 3 segundos.

**Reset\_delay\_15m**: Señal digital de salida del mss para resetear el contador del delay correspondiente a 15 min.

**Reset\_delay\_3s**: Señal digital de salida del mss para resetear el contador del delay correspondiente a 3 segundos.

**toldo\_replegado:** Señal digital que indica cuando el toldo esta replegado.

**toldo\_desplegado:** Señal digital que indica cual el toldo esta desplegado.

1. Código VHDL

**Controlador MSS**

En el código definimos los process en los que se describe el decodificador de estados siguientes, y el decodificador de estados presentes, se definen las señales de entrada, salida e internas y se describe estado se establece con los valores que toman las entradas.

library ieee;

use ieee.std\_logic\_1164.all;

entity controlador\_mss is

port( clock,reset,start,termostato,anemometro,sensor\_lluvia,sensor\_ventana,ok\_delay\_15m,

ok\_delay\_3s: in std\_logic;

replegar\_toldo,desplegar\_toldo,toldo\_desplegado,encender\_ventilador,

en\_delay\_3s,en\_delay\_15m,reset\_delay\_3s,reset\_delay\_15m: out std\_logic;

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

end controlador\_mss;

architecture solucion of controlador\_mss is

type estados is (Ta,Tb,Tc,Td,Te,Tf,Tg,Th,Ti,Tj,Tk,Tl,Tm,Tn,Tp);

signal y: estados;

begin

process(clock,reset)

begin

if reset <= '0' then y <= Ta;

elsif clock'event and clock = '1' then

case y is

when Ta => if start='1' then y <= Tb; else y <= Ta; end if;

when Tb => if start ='0' then y <= Tc; else y <= Tb; end if;

when Tc => if sensor\_lluvia ='0' and sensor\_ventana = '1' then y <= Td; else y <= Tf; end if;

when Td => if ok\_delay\_3s ='1' then y <= Te; else y <= Td; end if;

when Te => if sensor\_lluvia='1' then y <= Tc; else y <= Te; end if;

when Tf => if termostato='1' then y <= Tg;

elsif termostato = '0' and sensor\_lluvia = '1' and sensor\_ventana = '1' then y <= Tk;

else y <= Tn; end if;

when Tg => if ok\_delay\_3s ='1' then y <= Th; else y <= Tg; end if;

when Th => if ok\_delay\_15m ='1' and termostato ='1' then y <= Tm;

elsif ok\_delay\_15m ='0’ then y <= Th; else y <= Ti; end if;

when Ti => if ok\_delay\_3s ='1' then y <= Tj; else y <= Ti; end if;

when Tj => if (termostato or sensor\_lluvia)='1' then y <= Tc; else y <= Tj; end if;

when TK => if ok\_delay\_3s ='1' then y <= Tl; else y <= Tk; end if;

when Tl => if (termostato or not(sensor\_lluvia) or not(sensor\_ventana)) = '1'

then y <= Tc; else y <= Tl; end if;

when Tm => if termostato='0' then y <= Ti; else y <= Tm; end if;

when Tn => if ok\_delay\_3s='1' then y <= Tp; else y <= Tn; end if;

when Tp => if termostato = '1' then y <= Tc; else y <= Tp; end if;

end case;

end if;

end process;

process(y)

begin

replegar\_toldo <= '0'; desplegar\_toldo<= '0'; toldo\_desplegado <= '0'; encender\_ventilador <= '0'; en\_delay\_3s <= '0';en\_delay\_15m <= '0'; reset\_delay\_3s <= '0'; reset\_delay\_15m <= '0';

case y is

when Ta => estado <= "0000";reset\_delay\_3s <= '1'; reset\_delay\_15m<= '1';

when Tb => estado <= "0001";

when Tc => estado <= "0010"; reset\_delay\_15m<= '1';reset\_delay\_3s <= '1';

when Td => estado <= "0011";desplegar\_toldo <= '1'; en\_delay\_3s<= '1';

when Te => estado <= "0100";toldo\_desplegado<= '1';

when Tf => estado <= "0101";

when Tg => estado <= "0110";desplegar\_toldo <= '1'; en\_delay\_3s<= '1';

when Th => estado <= "0111";toldo\_desplegado <= '1';reset\_delay\_3s<= '1'; en\_delay\_15m<= '1';

when Ti => estado <= "1000"; replegar\_toldo <= '1'; en\_delay\_3s <= '1';

when Tj => estado <= "1001";

when Tk => estado <= "1010"; desplegar\_toldo <= '1'; en\_delay\_3S <= '1';

when Tl => estado <= "1011"; toldo\_desplegado <= '1';reset\_delay\_3s <= '1';

when Tm =>estado<= "1100"; toldo\_desplegado<= '1';encender\_ventilador<= '1';

when Tn => estado <= "1101"; replegar\_toldo <= '1'; en\_delay\_3s<= '1';

when Tp => estado <= "1110";

end case;

end process;

end solucion;

**Antirrebote**

Este bloque nos permite realizar la correcta lectura de los pulsadores y evitar el rebote que este produce cuando este esta siendo pulsado, con el anti-rebote se lee el botón como pulsado después de ser soltado.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.all;

USE IEEE.STD\_LOGIC\_ARITH.all;

USE IEEE.STD\_LOGIC\_UNSIGNED.all;

-- Debounce Pushbutton: Filters out mechanical switch bounce for around 40Ms.

ENTITY ANTIREBOTE IS

PORT(PB\_N, CLOCK\_100Hz : IN STD\_LOGIC;

PB\_SIN\_REBOTE : OUT STD\_LOGIC);

END ANTIREBOTE;

ARCHITECTURE a OF ANTIREBOTE IS

SIGNAL SHIFT\_PB : STD\_LOGIC\_VECTOR(3 DOWNTO 0);

BEGIN

-- Debounce clock should be approximately 10ms or 100Hz

PROCESS

BEGIN

WAIT UNTIL (clock\_100Hz'EVENT) AND (clock\_100Hz = '1');

-- Use a shift register to filter switch contact bounce

SHIFT\_PB(2 DOWNTO 0) <= SHIFT\_PB(3 DOWNTO 1);

SHIFT\_PB(3) <= NOT PB\_N;

IF SHIFT\_PB(3 DOWNTO 0)="0000" THEN

PB\_SIN\_REBOTE <= '0';

ELSE

PB\_SIN\_REBOTE <= '1';

END IF;

END PROCESS;

END a;

**Divisor de frecuencia**

Este bloque recibe el reloj del sistema y divide esta para frecuencias menores esto nos sirve para por ejemplo usar una frecuencia de 1Hz para realizar conteo de segundos. O un clock diferente para los bloques anti-rebotes.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.all;

USE IEEE.STD\_LOGIC\_ARITH.all;

USE IEEE.STD\_LOGIC\_UNSIGNED.all;

ENTITY CLOCK\_DIV\_50 IS

PORT

( CLOCK\_50MHz :IN STD\_LOGIC;

CLOCK\_1MHz :OUT STD\_LOGIC;

CLOCK\_100KHz :OUT STD\_LOGIC;

CLOCK\_10KHz :OUT STD\_LOGIC;

CLOCK\_1KHz :OUT STD\_LOGIC;

CLOCK\_100Hz :OUT STD\_LOGIC;

CLOCK\_10Hz :OUT STD\_LOGIC;

CLOCK\_1Hz :OUT STD\_LOGIC);

END CLOCK\_DIV\_50;

ARCHITECTURE a OF CLOCK\_DIV\_50 IS

SIGNAL count\_1Mhz: STD\_LOGIC\_VECTOR(5 DOWNTO 0);

SIGNAL count\_100Khz, count\_10Khz, count\_1Khz: STD\_LOGIC\_VECTOR(2 DOWNTO 0);

SIGNAL count\_100hz, count\_10hz, count\_1hz: STD\_LOGIC\_VECTOR(2 DOWNTO 0);

SIGNAL clock\_1Mhz\_int, clock\_100Khz\_int, clock\_10Khz\_int, clock\_1Khz\_int: STD\_LOGIC;

SIGNAL clock\_100hz\_int, clock\_10hz\_int, clock\_1hz\_int: STD\_LOGIC;

BEGIN

PROCESS

BEGIN

-- Divide by 50

WAIT UNTIL clock\_50Mhz'EVENT and clock\_50Mhz = '1'; -- 24 Mhz

IF count\_1Mhz < 49 THEN

count\_1Mhz <= count\_1Mhz + 1;

ELSE

count\_1Mhz <= "000000";

END IF;

IF count\_1Mhz < 4 THEN

clock\_1Mhz\_int <= '0';

ELSE

clock\_1Mhz\_int <= '1';

END IF;

-- Ripple clocks are used in this code to save prescalar hardware

-- Sync all clock prescalar outputs back to master clock signal

clock\_1Mhz <= clock\_1Mhz\_int;

clock\_100Khz <= clock\_100Khz\_int;

clock\_10Khz <= clock\_10Khz\_int;

clock\_1Khz <= clock\_1Khz\_int;

clock\_100hz <= clock\_100hz\_int;

clock\_10hz <= clock\_10hz\_int;

clock\_1hz <= clock\_1hz\_int;

END PROCESS;

-- Divide by 10

PROCESS

BEGIN

WAIT UNTIL clock\_1Mhz\_int'EVENT and clock\_1Mhz\_int = '1';

IF count\_100Khz /= 4 THEN

count\_100Khz <= count\_100Khz + 1;

ELSE

count\_100Khz <= "000";

clock\_100Khz\_int <= NOT clock\_100Khz\_int;

END IF;

END PROCESS;

-- Divide by 10

PROCESS

BEGIN

WAIT UNTIL clock\_100Khz\_int'EVENT and clock\_100Khz\_int = '1';

IF count\_10Khz /= 4 THEN

count\_10Khz <= count\_10Khz + 1;

ELSE

count\_10Khz <= "000";

clock\_10Khz\_int <= NOT clock\_10Khz\_int;

END IF;

END PROCESS;

-- Divide by 10

PROCESS

BEGIN

WAIT UNTIL clock\_10Khz\_int'EVENT and clock\_10Khz\_int = '1';

IF count\_1Khz /= 4 THEN

count\_1Khz <= count\_1Khz + 1;

ELSE

count\_1Khz <= "000";

clock\_1Khz\_int <= NOT clock\_1Khz\_int;

END IF;

END PROCESS;

-- Divide by 10

PROCESS

BEGIN

WAIT UNTIL clock\_1Khz\_int'EVENT and clock\_1Khz\_int = '1';

IF count\_100hz /= 4 THEN

count\_100hz <= count\_100hz + 1;

ELSE

count\_100hz <= "000";

clock\_100hz\_int <= NOT clock\_100hz\_int;

END IF;

END PROCESS;

-- Divide by 10

PROCESS

BEGIN

WAIT UNTIL clock\_100hz\_int'EVENT and clock\_100hz\_int = '1';

IF count\_10hz /= 4 THEN

count\_10hz <= count\_10hz + 1;

ELSE

count\_10hz <= "000";

clock\_10hz\_int <= NOT clock\_10hz\_int;

END IF;

END PROCESS;

-- Divide by 10

PROCESS

BEGIN

WAIT UNTIL clock\_10hz\_int'EVENT and clock\_10hz\_int = '1';

IF count\_1hz /= 4 THEN

count\_1hz <= count\_1hz + 1;

ELSE

count\_1hz <= "000";

clock\_1hz\_int <= NOT clock\_1hz\_int;

END IF;

END PROCESS;

END a;

**Delay 3 Segundos**

El bloque realiza un conteo hasta 3 a partir de una señal de enable y envia una señal después de 3 segundos indicando que estos han pasado, dentro del código se compara el conteo con el valor fijo de 3.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.all;

USE IEEE.NUMERIC\_STD.all;

ENTITY delay3s IS

GENERIC (width:POSITIVE:=2);

PORT (clk : IN std\_logic;

reset : IN std\_logic;

enable: IN std\_logic;

ok\_3S : OUT std\_logic);

END delay3s;

ARCHITECTURE arch1 OF delay3s IS

SIGNAL cnt : UNSIGNED(width-1 DOWNTO 0);

SIGNAL count : std\_logic\_vector(width-1 DOWNTO 0);

BEGIN

pSeq : PROCESS (clk, reset) IS

BEGIN

IF reset = '1' THEN

cnt <= (others => '0');

ELSIF clk'event AND clk='1' THEN

IF enable='1' THEN

cnt <= cnt + 1;

END IF;

END IF;

END PROCESS;

count <= std\_logic\_vector(cnt);

ok\_3s <= '1' when count = "11" else '0';

END arch1;

**Delay 15 minutos**

Este contador al igual que el delay 3 segundos, cuenta hasta 15\*60 segundos, el valor del contador es almacenado y al final comparado con 900 segundos luego envia una señal de completo.

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.all;

USE IEEE.NUMERIC\_STD.all;

ENTITY delay15m IS

GENERIC (width:POSITIVE:=10);

PORT ( clk : IN std\_logic;

reset : IN std\_logic;

enable: IN std\_logic;

ok\_15m : OUT std\_logic);

END delay15m;

ARCHITECTURE arch1 OF delay15m IS

SIGNAL cnt : UNSIGNED(width-1 DOWNTO 0);

SIGNAL count : std\_logic\_vector(width-1 DOWNTO 0);

BEGIN

pSeq : PROCESS (clk, reset) IS

BEGIN

IF reset = '1' THEN

cnt <= (others => '0');

ELSIF clk'event AND clk='1' THEN

IF enable='1' THEN

cnt <= cnt + 1;

END IF;

END IF;

END PROCESS;

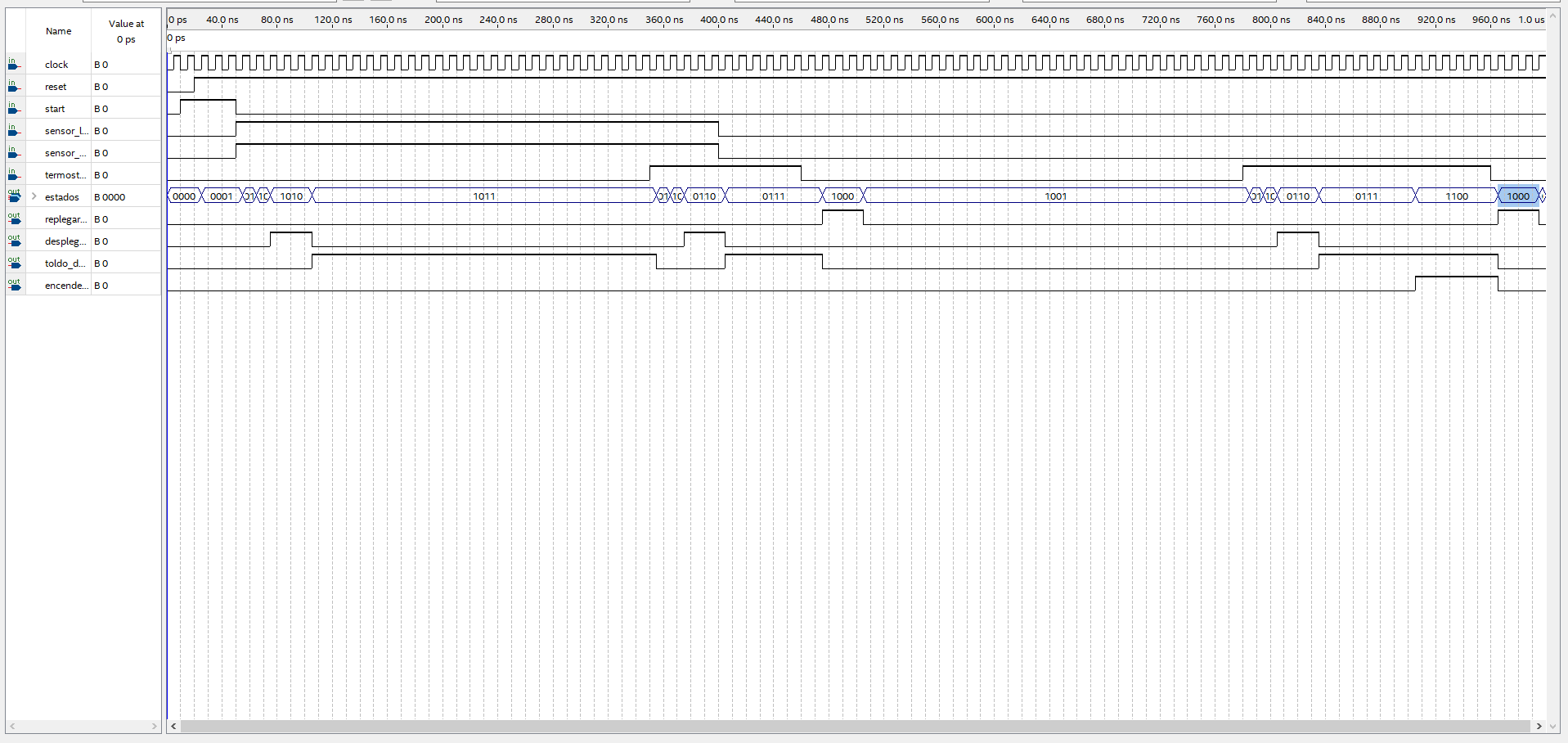
count <= std\_logic\_vector(cnt);

ok\_15m <= '1' when count = "1110000100" else '0';

END arch1;

1. Simulación WVF

Para la simulación correspondiente se han reemplazado el reloj de los bloques de delay de 3s y 15m por el clock del sistema esto para disminuir la espera de los estados, además se eliminan los antirrebotes.



1. Link de Video

<https://www.youtube.com/watch?v=jyV_tNanRaU&feature=youtu.be>