# TRABAJO SED: VHDL

*Raquel García Franco*
*Irene Álvarez Pérez*
*Lis Fortea Muñoz*

# ÍNDICE

# 1. INTRODUCCIÓN

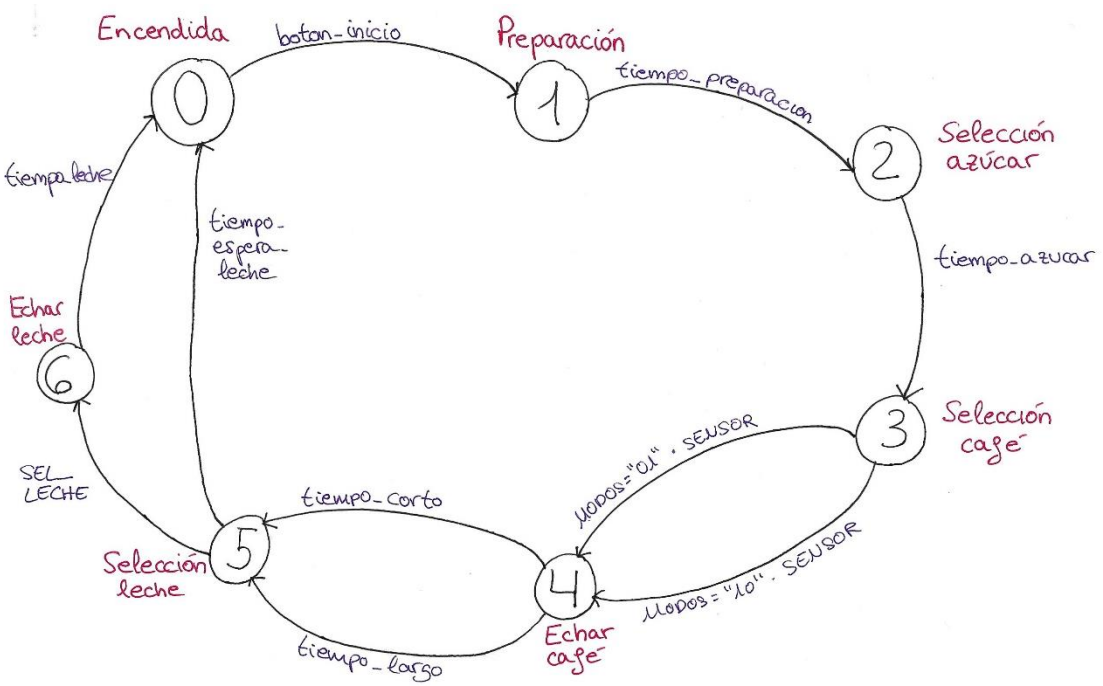
El presente trabajo trata del control de una cafetera, mediante la FPGA Nexys 4 DDR. La cafetera cuenta con dos modos para el café, uno corto de 10 segundos de duración y otro largo de 20 segundos. Además, se le ha incluido un depósito de leche y la opción de agregar azúcar. Más adelante se detallarán todos los elementos utilizados para la interfaz de usuario.

# 2. DESCRIPCIÓN DE ESTRATEGIA Y ALGORITMOS

Para la realización del código principal de control de las diferentes etapas de la cafetera se ha seguido el modelo de máquina de estado, de manera que contamos con dos FSM, una maestra que se encarga de gestionar los estados y una esclava encargada de los temporizadores para cada uno de ellos.

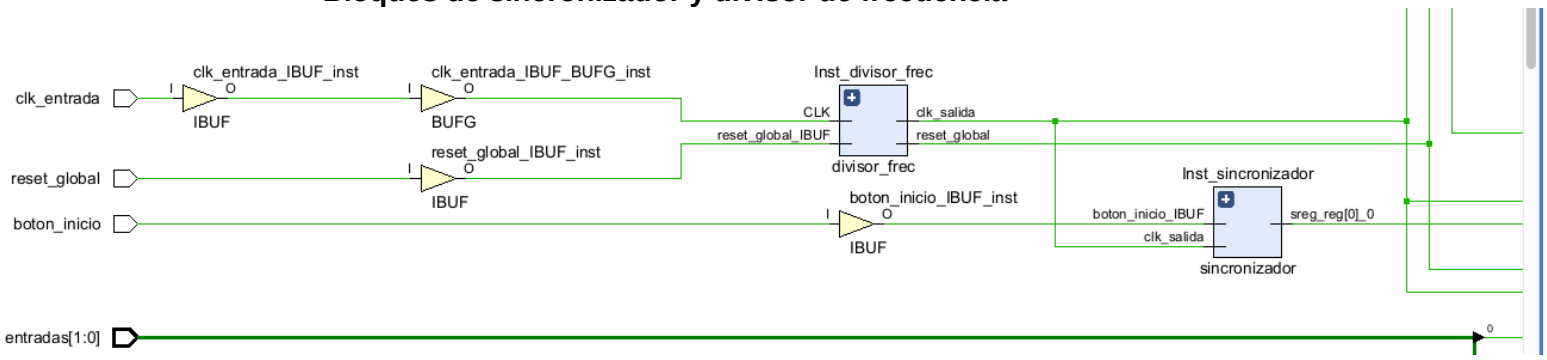
# 3. DIAGRAMAS

- DIAGRAMA DE ESTADOS

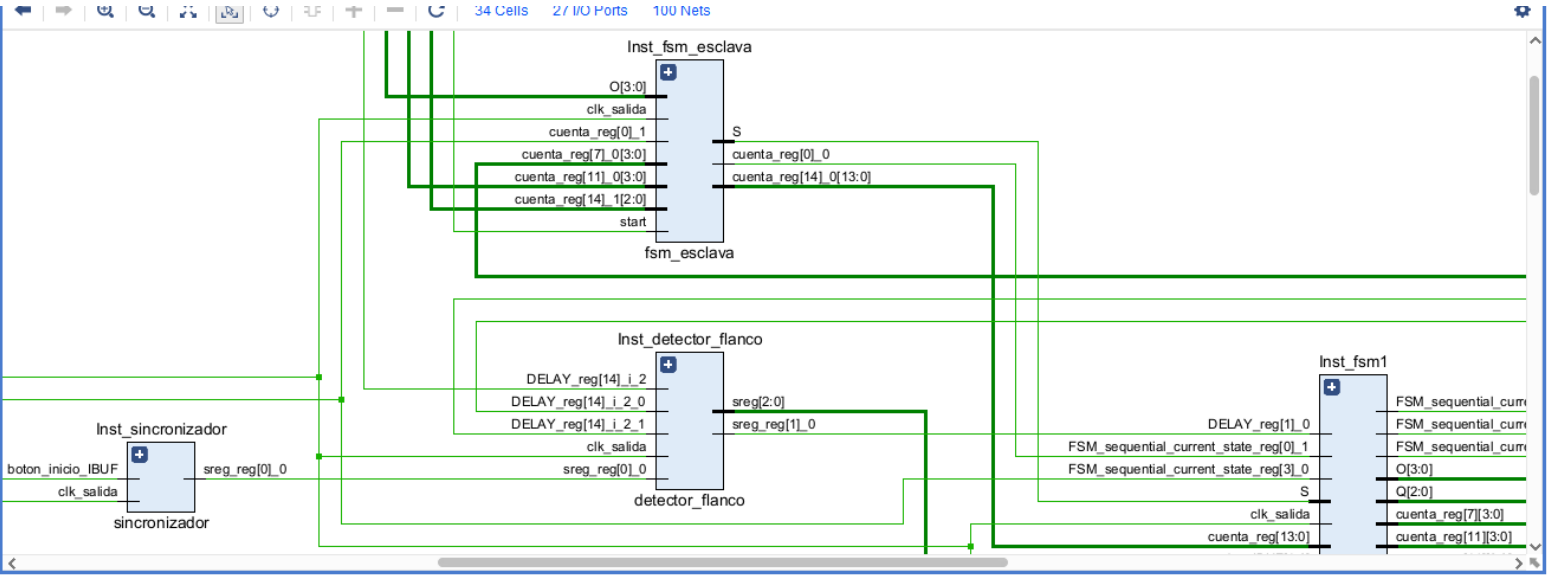


- DIAGRAMA DE BLOQUES DE DISEÑO

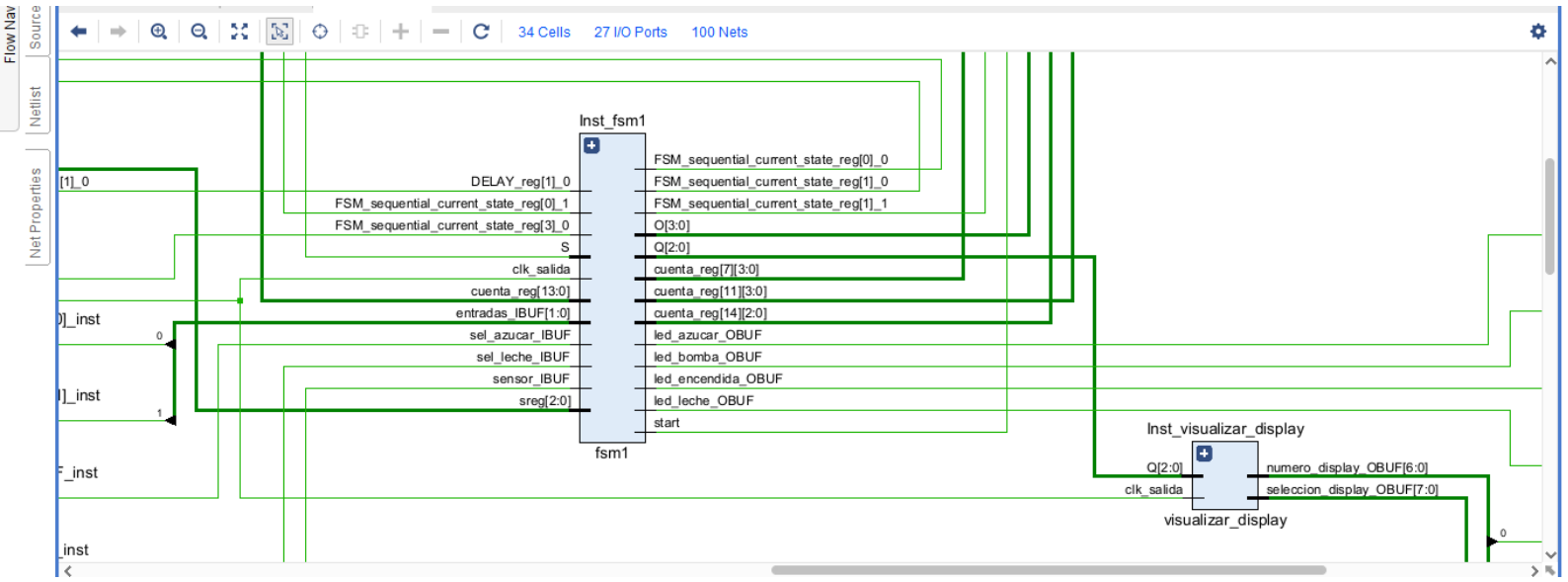
### - Bloques de sincronizador y divisor de frecuencia



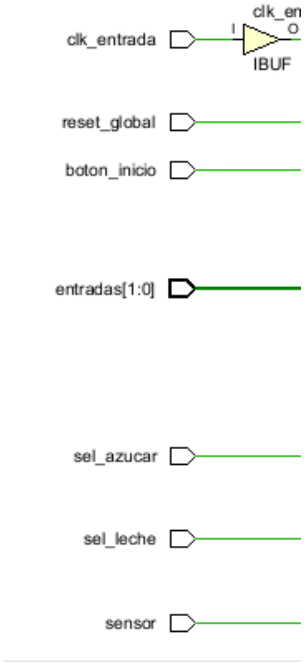

### - Bloques de fsm esclava y detector de flanco



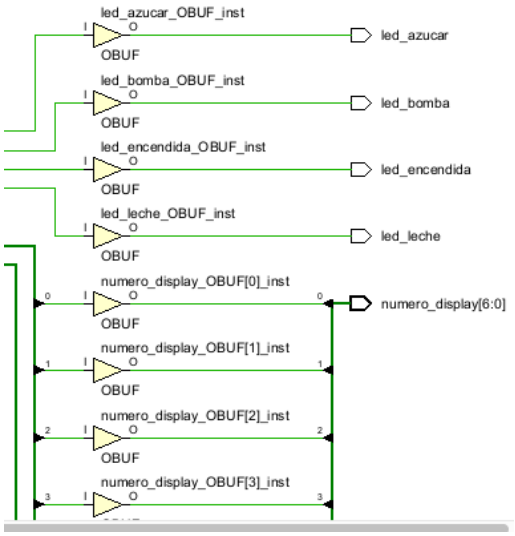

### - Bloques de fsm maestra y visualizar display

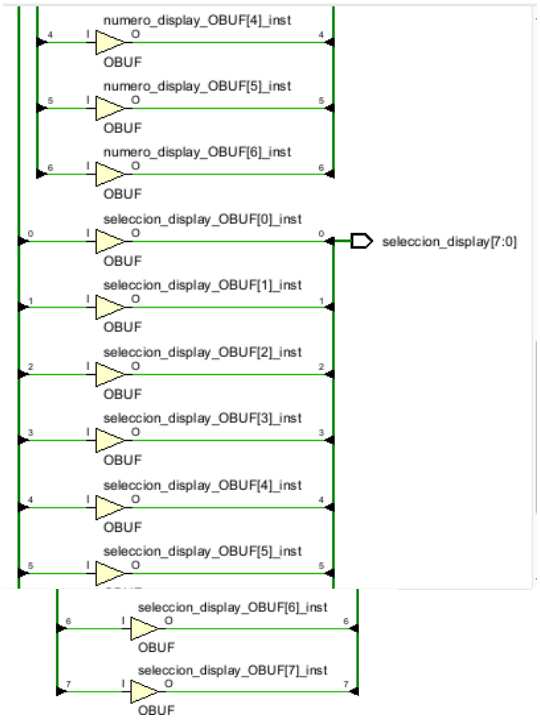


- **Entradas**

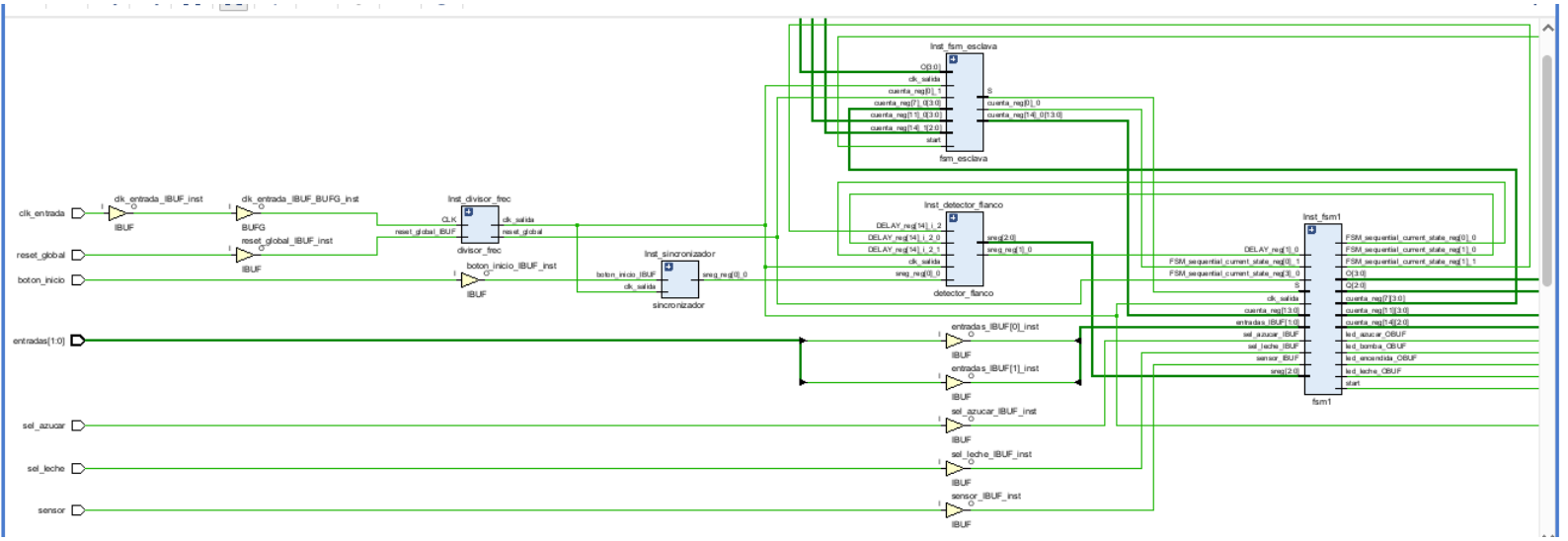


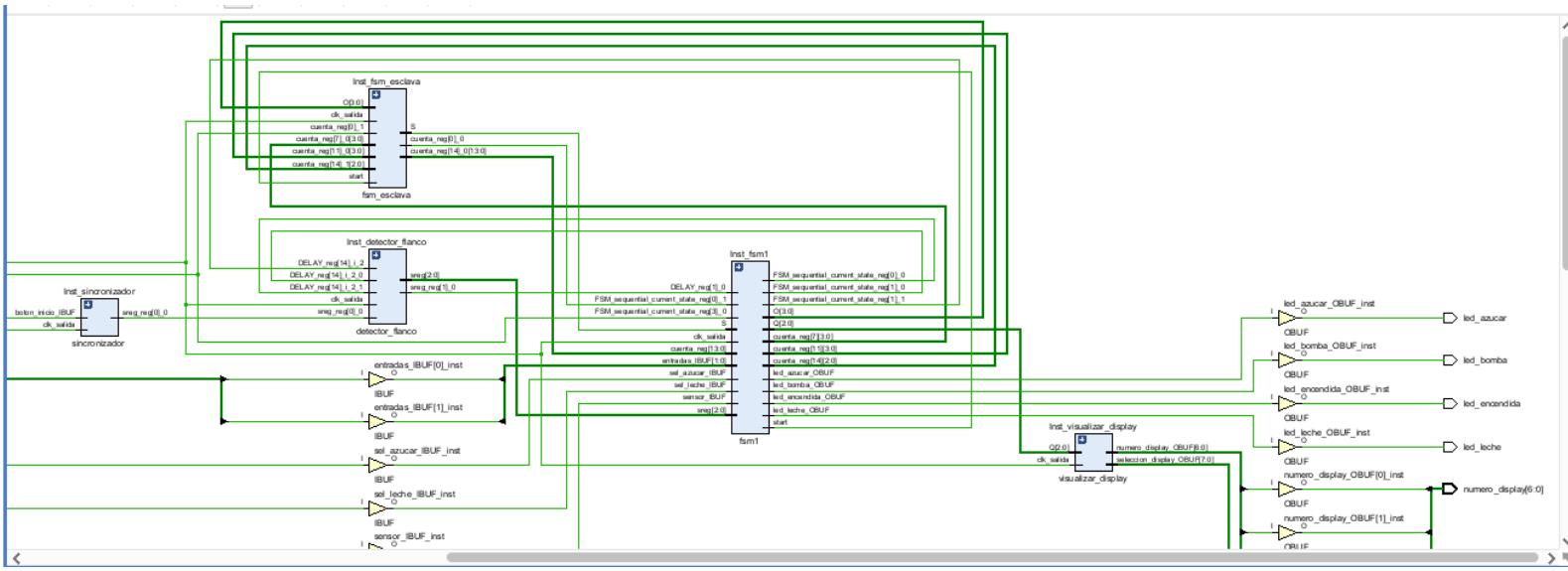

- **Salidas**






- **Diagrama de diseño completo**






# 4. DESCRIPCIÓN DE LOS ESTADOS

**ESTADO 0:** Estado de máquina encendida, la máquina está esperando la activación del botón de inicio para pasar al siguiente estado y comenzar a preparar el café.

**ESTADO 1:** Estado de máquina en preparación, se inicia el contador del tiempo de preparación de la cafetera y, transcurrido ese tiempo, se pasa al siguiente estado.

**ESTADO 2:** Estado de selección de azúcar, se inicia el contador del tiempo de selección de azúcar e, independientemente de la selección del usuario, una vez transcurrido este tiempo se pasa al siguiente estado.

**ESTADO 3:** Estado de selección del tipo de café, se pasa al siguiente estado cuando el usuario selecciona con los switches el modo que desea y coloca el vaso en el sensor.

**ESTADO 4:** Estado para echar el café, se activa la bomba y, transcurrido el tiempo determinado en función del modo seleccionado, se pasa al siguiente estado.

**ESTADO 5:** Estado de selección de leche, se inicial el contador del tiempo de selección de leche y se plantean dos posibilidades: transcurrido el tiempo sin que el usuario active el switch para indicar que desea leche, se volverá al estado de reposo o, en caso de que se active el switch, se pasará al estado 6.

**ESTADO 6:** Estado para echar la leche, se activa la bomba y, transcurrido el tiempo determinado para echar la leche, se vuelve al estado de reposo.

# 5. DESCRIPCIÓN DE LOS BLOQUES DE DISEÑO

**SINCRONIZADOR:** Dado que se utiliza un botón para el encendido de la cafetera, es necesario controlar los rebotes de éste ya que si no se podría interpretar como varias pulsaciones. Para ello, es necesario este bloque de sincronización que evita la metaestabilidad y permite que se detecte una sola pulsación.

**DETECTOR DE FLANCO:** Cuando se pulsa el botón se genera un pulso de duración indeterminada. Para solucionar esto utilizamos este bloque, el cual genera un solo pulso cada vez que detecta un flanco de bajada en la señal de entrada (botón). De esta forma, cada vez que pulsemos el botón de inicio se generará una señal que dura un ciclo de reloj, y consecuentemente sólo habrá un flanco de subida en un período de reloj.

**DIVISOR DE FRECUENCIA:** Este bloque ha sido utilizado para ajustar la frecuencia de nuestro circuito y coordinar los tiempos para que se asemejen a la realidad.

**FSM MAESTRA:** Este bloque se trata de una máquina de estados maestra que controla la evolución de las distintas etapas de nuestra cafetera, determinando las condiciones y llevando a cabo las órdenes necesarias para la activación de las diferentes salidas y de los temporizadores.

**FSM ESCLAVA:** Se trata de una máquina de estados esclava que se comunica con la maestra, encargándose de realizar las cuentas correspondientes a los distintos temporizadores que permiten pasar de estado. De esta manera, la maestra es la que ejecuta las órdenes de activación de los temporizadores y la esclava es la que una vez acabada la cuenta se lo indica a su maestra.

**DECODIFICADOR:** Un decodificador es un circuito combinacional que realiza la operación inversa a la de un codificador de datos, por eso también se puede definir un decodificador como circuito que convierte un código binario concreto en una forma sin codificar. De esta manera, en este caso se ha utilizado este bloque para poder utilizar el display de 7 segmentos, recibiendo diferentes instrucciones en binario a partir de las cuales el display recibe la orden de escribir determinadas palabras durante el proceso.

**VISUALIZAR DISPLAY:** Este bloque de diseño permite el encendido de cada uno de los displays de los que dispone la FPGA.

# 6. EXPLICACIÓN DE INTERFAZ Y FUNCIONAMIENTO

Nuestra cafetera consta de los siguientes elementos para su control y funcionamiento:

- ***Un botón de inicio***: Encargado de indicar a la cafetera que se desea preparar un café y que comience a prepararse para funcionar.
- ***5 switches***: Los dos primeros destinados a los modos (uno para seleccionar el modo corto y otro para el largo), uno para seleccionar si se desea azúcar, otro para indicar que queremos que lleve leche y, por último, un sensor de presencia para que comience a funcionar la bomba de café, una vez elegido el modo, cuando se coloque el vaso.
- ***4 LEDS*:** El primero de ellos nos indicará en el primer estado que la cafetera ha comenzado a preparar el café, el siguiente que sólo se iluminará si el usuario selecciona que quiere azúcar, otro para indicar que la bomba está en funcionamiento, ya sea echando café o leche, y el último que se iluminará cuando se desee leche en el café.
- ***Botón de RESET***: Se utilizará en el caso de que se desee resetear nuestra máquina. En el caso de ser pulsado se volverá al estado de encendida por defecto de la cafetera.
- ***Displays***: Para nuestro trabajo se han utilizado los 8 displays disponibles en la FPGA. Estos se encargan de indicar al usuario en que etapa de selección se encuentra, aparecerá la palabra “ON” en el estado de máquina encendida, durante el estado de preparación aparecerá “PREPARAR” ,en el siguiente estado “AZÚCAR” cuando se deba seleccionar esta, en el estado de selección de café nos indicará con la palabra “ELIGE” que seleccionemos el modo y aparecerá “CORTO” o “LARGO” en función del que se desee para que, una vez confirmada la selección, aparezca el tiempo para cada uno (“10 o 20 SEGS”) y, por último, mostrará si se ha elegido que lleva leche visualizando “LECHE” o “NO LECHE”.

De esta manera, el funcionamiento procederá de la siguiente manera:

1. ***Pulsación del botón de inicio****:* Primeramente, el usuario pulsará el botón de encendido que hará que la cafetera pase de su estado de reposo al de preparación. También se encenderá un led que nos indica que está en funcionamiento.
2. ***Selección de azúcar****:* Pasado el tiempo de preparación (5 segundos), el display indicará que se debe elegir azúcar, si el usuario activa el switch correspondiente se encenderá el led de selección de azúcar. Una vez pasado el tiempo reservado para esta decisión (5 segundos), se pasará al siguiente estado para elegir el café.
3. ***Selección del modo de café****:* El usuario dispondrá de dos switches, uno para cada modo. Dependiendo del modo que se elija, el display mostrará nuestra elección. A continuación, la cafetera no empezará a echar el café hasta que se coloque el vaso en el sensor de presencia (simulado con un switch), tras esto, se encenderá el led de la bomba, el display nos indicará el tiempo según el modo (10 o 20 segundos) y transcurrido este, pasaremos al estado de selección de leche.
4. ***Selección de leche****:* En este estado el usuario dispondrá de otro switch para indicar si desea o no leche, encendiéndose un led en caso afirmativo y en ambos casos mostrando por display nuestra elección. En caso de desear leche, se activará la bomba con su led correspondiente y transcurridos 10

segundos se dará por terminado el proceso de preparación de nuestro café y se volverá al estado de reposo.

# 7. FUNCIONES IMPORTANTES

- FSM MAESTRA

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;


entity fsm1 is
    generic (
        long_opcion:positive:=4
    );
    port (
        RESET : in std_logic; --reset asíncrono
        CLK : in std_logic; --reloj activo en el flanco de subida
        EDGE : in std_logic; --indicación de que se ha pulsado el boton de inicio

        MODOS : in std_logic_vector(1 downto 0); --switches para los modos del cafe
        SEL_LECHE: in std_logic; --switch para seleccionar si se desea leche
        SEL_AZUCAR: in std_logic; --switch para seleccionar si se desea azúcar
        SENSOR: in std_logic; --switch que simula sensor de presencia

        DONE: in std_logic; --entrada recibida de la esclava indica que se ha terminado
de contar

        LED_ENCENDIDA: out std_logic; --led que muestra que se va a comenzar a
prepar cafe
        LED_BOMBA: out std_logic; --led indica que esta activa la bomba
        LED_LECHE: out std_logic; --led indica que se desea leche
        LED_AZUCAR: out std_logic; --led indica que se desea azucar
        START: out std_logic;
        MODO_DISPLAY: out std_logic_vector(long_opcion -1 downto 0);--codigo que
indica al display que palabra msotrar
        DELAY : out unsigned (14 downto 0)
    );
end fsm1;

architecture Behavioral of fsm1 is

    type STATES is (S0, S1, S2_1, S2_2, S3_1, s3_2, S4, S5_1, s5_2, S6);
    signal current_state: STATES := S0;
    signal next_state: STATES;
    constant tiempo_preparacion :positive := 5000; --tiempo de calentamiento/molido
cafe
    constant tiempo_azucar : positive := 5000; --tiempo de seleccion de azucar
    constant tiempo_corto : positive := 12000;  --tiempo de echar el cafe corto
    constant tiempo_largo : positive:= 21000;--tiempo de echar cafe largo
    constant tiempo_espera_leche: positive := 5000; --tiempo de seleccion leche
    constant tiempo_leche : positive := 11000; --tiempo de echar la leche
```

```
begin

   state_register: process (RESET, CLK)
   begin
      if RESET = '0' then
         current_state <= S0; --vuelve al estado por defecto encendida

      elsif rising_edge(CLK) then
         current_state <= next_state; --se pasa al siguiente estado
      end if;
   end process;

   nextstate: process (RESET, MODOS,EDGE, current_state,
DONE,SEL_LECHE,SEL_AZUCAR,SENSOR,MODOS)
   begin
      next_state <= current_state;
      case current_state is
         when S0 => --estado de maquina encendida

            if EDGE = '1' then --se pulsa el boton de inicio
               next_state <= S1; --pasa a preparación

            end if;

         when S1 => --estado de preparación

            if DONE = '1' then --terminada la cuenta del tiempo se pasa al siguiente
estado
               next_state <= S2_1;
            end if;

         when S2_1=> --estado para cargar el tiempo de seleccion azúcar

            next_state <= S2_2;

         when s2_2=> --estado de seleccion azucar

            if DONE = '1' then --pasado el tiempoo de seleccion de azucar se va al
estado siguiente
               next_state <= S3_1;
            end if;

         when S3_1 => --estado para cargar tiempos del café

            if MODOS="01" then --se elige modo corto
               next_state <= S3_2;

            end if;

            if MODOS="10" then

               next_state <= S3_2;
            end if;

         when s3_2 => --estado para detectar vaso
```

```
            if SENSOR='1' then --detecta que se coloca el vaso y pasa al estado de
echar cafe
                next_state<=s4;
            end if;

        when S4 => --estado echar cafe

            if DONE = '1' then --termina de contar el tiempo y pasa de estado
                next_state <= S5_1;
            end if;

        when S5_1 => --estado para cargar el tiempo de seleccion leche

            next_state <= S5_2;

        when S5_2 => --estado para seleccionar leche

            if SEL_LECHE ='1' then--se quiere leche
                next_state <= S6;-- se pasa a echar leche
            end if;

            if DONE = '1' then --si se termina el tiempo de seleecion sin querer leche se
vuelve a encendida
                next_state <= S0;
            end if;

        when S6 =>--estado para echar leche
            if DONE = '1' then --se termina de contar el tiempo de echar leche y se
vuelve a encendida
                next_state <= S0;
            end if;

      end case;
   end process;

   outputs: process (MODOS,EDGE, current_state,
SEL_LECHE,SEL_AZUCAR,SENSOR,MODOS)
case current_state is
        when S0 => --estado de maquina encendida
            LED_LECHE<='0'; --apaga todos los leds
            LED_BOMBA<='0';
            LED_AZUCAR <= '0';
            LED_ENCENDIDA <= '0';
            MODO_DISPLAY <= "0001"; --le indica al display que ponga "ON"
            if EDGE = '1' then --se pulsa el boton de inicio
                START <= '1';
                DELAY <= to_unsigned(tiempo_preparacion -2, DELAY'length); --carga el
tiempo de preparacion
            end if;

        when S1 => --estado de preparacion
            START<='0';
            DELAY<=(others=>'0');
            LED_ENCENDIDA <= '1'; --indica led que esta preparando
```

```
            MODO_DISPLAY<="0000"; --indica que aparece palabra "PREPARAR"


         when S2_1=> --estado para cargar el tiempo de seleccion azucar
            LED_ENCENDIDA <= '0';
            START<='1';
            DELAY <= to_unsigned(tiempo_azucar -2, DELAY'length); --carga el tiempo
de seleccion azucar
            MODO_DISPLAY <= "0010"; --indica que aparezca la plabra "AZUCAR"


         when s2_2=> --estado de seleccion azucar
            START<='0';
            DELAY<=(others=>'0');
            LED_ENCENDIDA <= '0';
            MODO_DISPLAY <= "0010"; --indica que aparezca la plabra "AZUCAR"
            if SEL_AZUCAR = '1' then --se selecciona que se desea azucar
               LED_AZUCAR <= '1'; --se actva led azucar
            end if;

         when S3_1 => --estado para cargar tiempos del cafe
            START <= '1';
            LED_BOMBA<='0';
            LED_ENCENDIDA <= '0';
            MODO_DISPLAY <= "0011"; --indica que escriba palabra "ELIGE"
            if MODOS="01" then --se elige modo cafe
               MODO_DISPLAY <= "0100"; --le dice al display que escriva "CORTO"
               DELAY <= to_unsigned(tiempo_corto -2, DELAY'length);--carga el tiempo
de corto
            end if;
            if MODOS="10" then
               MODO_DISPLAY <= "1000"; --le dice al display que escriba "LARGO"
               DELAY <= to_unsigned(tiempo_largo -2, DELAY'length);--carga el tiempo
de largo

            end if;

         when s3_2 => --estado para detectar vaso

            LED_ENCENDIDA <= '0';
            LED_BOMBA<='0';


         when S4 => --estado echar cafe
            LED_ENCENDIDA <= '0';
            LED_BOMBA <= '0';
            START <= '0';
            DELAY<=(others=>'0');
            if MODOS = "01" then
               LED_BOMBA <= '1'; --se activa la bomba
               MODO_DISPLAY <= "0110"; --se indica al display que indique el tiempo
corto "10 SEG"
            end if;

            if MODOS = "10" then
```

```
                LED_BOMBA <= '1'; --se activa la bomba
                MODO_DISPLAY <= "1010"; --se indica al display que indique el tiempo
largo "20 SEG"
            end if;


        when S5_1 => --estado para cargar el tiempo de seleccion leche
            LED_BOMBA<='0';
            LED_ENCENDIDA <= '0';
            START<='1';
            DELAY <= to_unsigned(tiempo_espera_leche -2, DELAY'length);--carga el
timpo de eleccion leche


        when S5_2 => --estado para seleccionar leche
            LED_BOMBA<='0';
            LED_ENCENDIDA <= '0';
            START <= '0';
            DELAY<=(others=>'0');

            if SEL_LECHE = '0' then --no se quiere leche
                MODO_DISPLAY <= "1110";--se indica al display que ponga "NO LECHE"
            end if;

            if SEL_LECHE ='1' then--se quiere leche
                LED_LECHE <= '1';--indica que queremos leche
                LED_BOMBA<='1';--activa la bomba
                MODO_DISPLAY <= "1100";--se indica al display que ponga "LECHE"
                START <= '1';
                DELAY <= to_unsigned(tiempo_leche -2, DELAY'length);--carga el tiempo
de echar leche
            end if;



        when S6 =>--estado para echar leche
            LED_ENCENDIDA <= '0';
            START<='0';
            DELAY<=(others=>'0');
            LED_LECHE <= '1';
            LED_BOMBA<='1';--se activa la bomba
            end if;

        when others =>
            MODO_DISPLAY<= "0000";
            LED_LECHE<='0';
            LED_BOMBA<='0';
            LED_AZUCAR <= '0';
            LED_ENCENDIDA <= '0';
            start<='0';
            DELAY<=(others=>'0');
    end case;


end Behavioral;
```

- ## FSM ESCLAVA

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

-- Implementamos un timer
entity fsm_esclava is
port(
    CLK     : in std_logic; --señal de reloj
    RESET   : in std_logic; --reset activo a nivel alto
    START   : in std_logic; -- señal de inicio
    DELAY   : in unsigned (14 downto 0); -- tiempo de espera
    DONE    : out std_logic --señal de fin

);
end fsm_esclava;

architecture Behavioral of fsm_esclava is
  signal cuenta : unsigned (DELAY'range);

begin
    process(RESET, CLK)
    begin
    if RESET = '0' then --si pulsamos el reset ponemos todo a 0
      cuenta <=(others => '0');

    elsif rising_edge(CLK) then
      if START ='1' then
        cuenta <= DELAY; --se carga el valor de delay en cuenta
      elsif cuenta /= 0 then
        cuenta <= cuenta -1;

      end if;
    end if;
  end process;
    DONE <= '1' when cuenta = 1 else '0'; --vale '1' cuando se ha acabado la
cuenta

end Behavioral;
```

- ## DECODIFICADOR

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_unsigned.ALL;
ENTITY decodificador IS
   generic (
      long_opcion:positive:=4
   );
   PORT (
      seleccion : in std_logic_vector(long_opcion -1 downto 0);
      salida_disp0 : OUT std_logic_vector(6 downto 0);
```

```
        salida_disp1: out std_logic_vector(6 downto 0);
        salida_disp2 : out std_logic_vector(6 downto 0);
        salida_disp3 : out std_logic_vector(6 downto 0);
        salida_disp4 : out std_logic_vector(6 downto 0);
        salida_disp5 : out std_logic_vector(6 downto 0);
        salida_disp6 : out std_logic_vector(6 downto 0);
        salida_disp7 : out std_logic_vector(6 downto 0)
    );
END ENTITY decodificador;
ARCHITECTURE behavioral OF decodificador IS
BEGin
    process(seleccion)
    begin
        case seleccion is
            when "0000"=>
                salida_disp7<="0000110";--E
                salida_disp6<="0010010";--S
                salida_disp5<="0001100";--P
                salida_disp4<="0000110";--E
                salida_disp3<="1001110";--r
                salida_disp2<="0001000";--A
                salida_disp1<="1111111";---
                salida_disp0<="1111111";---
            when "0010"=> --Nivel azúcar> AZUCAR
                salida_disp7<="0001000";--A
                salida_disp6<="0100100";--Z
                salida_disp5<="1000001";--U
                salida_disp4<="1000110";--C
                salida_disp3<="0001000";--A
                salida_disp2<="1001110";--r
                salida_disp1<="1111111";--vacio
                salida_disp0<="1111111";--vacio
            when "0100"=> --corto
                salida_disp7<="1000110";--C
                salida_disp6<="1000000";--o
                salida_disp5<="1001110";--r
                salida_disp4<="0000111";--t
                salida_disp3<="1000000";--o
                salida_disp2<="1111111";--vacio
                salida_disp1<="1111111";--vacio
                salida_disp0<="1111111";--vacio
            when "0110"=> --corto-10seg
                salida_disp7<="1111001";--1
                salida_disp6<="1000000";--0
                salida_disp5<="0010010";--S
                salida_disp4<="0000110";--E
                salida_disp3<="0000010";--G
                salida_disp2<="1111111";--vacio
                salida_disp1<="1111111";--vacio
                salida_disp0<="1111111";--vacio
            when "1000"=> --largo
                salida_disp7<="1000111";--L
                salida_disp6<="0001000";--A
                salida_disp5<="1001110";--r
                salida_disp4<="0000010";--G
```

```
    salida_disp3<="1000000";--0
    salida_disp2<="1111111";--vacio
    salida_disp1<="1111111";--vacio
    salida_disp0<="1111111";--vacio
when "1010"=> --largo-20seg
    salida_disp7<="0100100";--2
    salida_disp6<="1000000";--0
    salida_disp5<="0010010";--S
    salida_disp4<="0000110";--E
    salida_disp3<="0000010";--G
    salida_disp2<="1111111";--vacio
    salida_disp1<="1111111";--vacio
    salida_disp0<="1111111";--vacio
when "1100"=> -- con leche
    salida_disp7<="1000111";--L
    salida_disp6<="0000110";--E
    salida_disp5<="1000110";--C
    salida_disp4<="0001001";--H
    salida_disp3<="0000110";--E
    salida_disp2<="1111111";--vacio
    salida_disp1<="1111111";--vacio
    salida_disp0<="1111111";--vacio
when "1110"=> -- sin leche
    salida_disp7<="1001000";--N
    salida_disp6<="1000000";--O
    salida_disp5<="1111111";--vacio
    salida_disp4<="1000111";--L
    salida_disp3<="0000110";--E
    salida_disp2<="1000110";--C
    salida_disp1<="0001001";--H
    salida_disp0<="0000110";--E
  when "0001"=> -- sin leche
    salida_disp7<="1000000";--O
    salida_disp6<="1001000";--N
    salida_disp5<="1111111";--vacio
    salida_disp4<="1111111";--vacio
    salida_disp3<="1111111";--vacio
    salida_disp2<="1111111";--vacio
    salida_disp1<="1111111";--vacio
    salida_disp0<="1111111";--vacio
     when "0011"=> -- sin leche
    salida_disp7<="0000110";--E
    salida_disp6<="1000111";--L
    salida_disp5<="1001111";--I
    salida_disp4<="0000010";--G
    salida_disp3<="0000110";--E
    salida_disp2<="1111111";--vacio
    salida_disp1<="1111111";--vacio H
    salida_disp0<="1111111";--vacio
when others=>
    salida_disp7<="1111111";--vacio
    salida_disp6<="1111111";--vacio
    salida_disp5<="1111111";--vacio
    salida_disp4<="1111111";--vacio
    salida_disp3<="1111111";--vacio
```

```
            salida_disp2<="1111111";--vacio
            salida_disp1<="1111111";--vacio
            salida_disp0<="1111111";--vacio
        end case;
    end process;
end behavioral;
```

# 8. BIBLIOGRAFÍA

Guiones de prácticas de laboratorio y diapositivas de la asignatura Sistemas Electrónicos Digitales.