Memoria

Sistemas Electrónicos para la Automatización

Docente: *Jorge Romero Sánchez*

Trabajo Final de la asignatura: Análisis, Simulación y Síntesis de un PLC

***Izan Amador Bustos**: izan.amador@uma.es*

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Contenido

Lista de figuras

Lista de códigos

1. Especificaciones

- **Estudio previo:** Describid en detalle el PLC cuyo código se suministra, y que no puede ser modificado una vez seleccionado el estilo de FSM: Moore o Mealy, que elegisteis en la 2ª práctica.
- Como parámetros fijos el sistema tendrá k_Max = 3 entradas, p_Max = 4 salidas y m_Max = 4 biestables tal y como aparece en el package *Tipos_FSM_PLC*. O de otra forma, con independencia de la FSM que se programe, todos tendréis que simular y sintetizar a nivel RTL el PLC con *k=3(=k_Max)* entradas aunque vuestra FSM de ejemplo tenga *k=1* entrada; *p=4(=p_Max)* salidas aunque vuestra FSM de ejemplo tenga *p=2* salidas; y *m=4(=m_Max)* biestables aunque vuestra FSM de ejemplo tenga *m=2* biestables.
- Generar un único proyecto en Vivado que implemente dicho PLC.
- Para la fase de simulación diseñad un único Test Bench con manejo de ficheros (modelo en Tema 5) que ilustre el funcionamiento correcto del sistema con los parámetros fijos para todos k=k_Max, p=p_Max y m=m_Max: para ello se programarán en las tablas, la FSM de la 2ª práctica "Diseño, Simulación y Síntesis de una FSM en VHDL" que os corresponda.
- Generad los rebotes en la señal de Trigger como si fuese un conmutador externo de Pull-Up desde un proceso. El único código VHDL que habrá que incluir en la memoria será el del Test Bench diseñado, comentado y correctamente tabulado: cuidad el estilo de programación.
- **Sintetizad** el PLC a nivel RTL (RTL Analysis) para *k=k_Max, p=p_Max y m=m_Max*, para comprobar que efectivamente se corresponde con lo que representa.

2. Estudio del código - FSM Moore

Para la realización de la práctica, se suministran los siguientes componentes, los cuales serán analizados en esta sección.

2.1. CKE_Gen

La funcionalidad de este módulo es la generación de una señal de pulso de reloj con ciclo de duración a partir de la salida del módulo Debouncer.

2.2. Debouncer

La finalidad de este componente es eliminar las rápidas variaciones que se producen en una señal de entrada al pulsar el botón, derivadas de la naturaleza mecánica del sistema, obteniendo únicamente los cambios de estado del mismo.

2.3. Reg_Des

Se utiliza para almacenar una señal en la entrada del sistema, dentro del componente *Sincronizador* para determinar si ha ocurrido un cambio de estado en base a los valores almacenados en el mismo. Posee un tamaño de filtro variable ajustable dependiendo de las características mecánicas del dispositivo.

2.4. Sincronizador

Se compone de los módulos de *CKE_Gen*, *Debouncer* y *Reg_Des*, para formar un elemento sincronizador que permite el correcto funcionamiento de dispositivos físicos como botones.

2.5. MUX_PLC

Se trata de un multiplexor programable para PLC, que selecciona un valor de una tabla ROM en función de una dirección de entrada y lo asigna a la salida después de un tiempo específico.

2.6. Reg_PLC

En este módulo se implementa un registro de desplazamiento que desplaza a los bits hacia la derecha cuando se activa la entrada de control des y se actualiza en el flanco de subida del reloj.

2.7. Mis_Tipos_PLC

En este paquete se define un tipo y una serie de constantes comunes para la implementación del PLC. Se define el tipo $Tabla\_FSM$ que se utiliza para generar una ROM, y las constantes N_Bits_Dato, K_Max, p_Max, y m_Max se utiillizan para especificar el número máximo de biestables en la FSM y los requisitos de entrada y salida del PLC.

2.8. FSM_PLC

Entidad
La salida es un vector que se corresponde con la salida de la FSM codificada, en nuestro caso, con 3 ceros más por la izquierda. Las entradas se describen a continuación:

- **X**: Entrada del diseño de la FSM.
- **Tabla de Estado**: almacena la codificación de los estados de la FSM tipo Moore. Se definirá en el banco de pruebas.
- **Tabla de salida**: almacena la codificación de las salidas para los distintos estados.
- **Entradas de control**: Clk, cke, $reset$, $Trigger$.

Componentes

En la **arquitectura**, se definen los tres componentes necesarios para la composición del sistema el *Mux_PLC*, *Reg_PLC*, y el *Sincronizador*.

Definición de señales
Se definen diferentes señales para la interconexión de los componentes, entre ellas: los *Estados y las salidas con formato*, los *datos intermedios*, el *estado actual* y *estado próximo* y la *salida del multiplexor* y del sistema.

Procesos de verificación de señales
Además se definen diferentes procesos pasivos para la comprobación de las especificaciones del enunciado:

- **Comprueba_2kxm** y **Comprueba_2kxp**: verifican las restricciones de tamaño.
- **Comprueba_TSUH**: verifica el tiempo de setup y de hold.
- **Comprueba_TW**: verifica la anchura de un pulso.

Procesos de la FSM
Se definen tres procesos *Sincronizador*, *Clock_enable*, *Reg_PLC* para el funcionamiento del sistema.

Ecuación de transición de estado
Se define el proceso *Asignacion_Mux*, para la asignación de los datos intermedios a los estados con formato.

La *Ecuacion_De_Transicion_1* selecciona a partir del estado actual todos los próximos posibles estados, mientras que la *Ecuacion_De_Transicion_2* selecciona a partir de la entrada actual el próximo estado.

El proceso *Asignacion_Proximo_Estado* sirve para ajustar el tamaño del dato del MUX de *N_Bits_Dato* a *m* bits del estado.

Finalmente los procesos de *C* y *Salida_Moore*, sirven para seleccionar la salida a partir del estado actual y proporcionar la salida de moore, respectivamente.

2.9. Verificación en el RTL

Puesto que el código perteneciente a los componentes de la practica se encuentra escrito a excepción del banco de pruebas, se realiza la síntesis del sistema satisfactoriamente y se pueden observar las conexiones de los distintos componentes en el esquemático RTL.

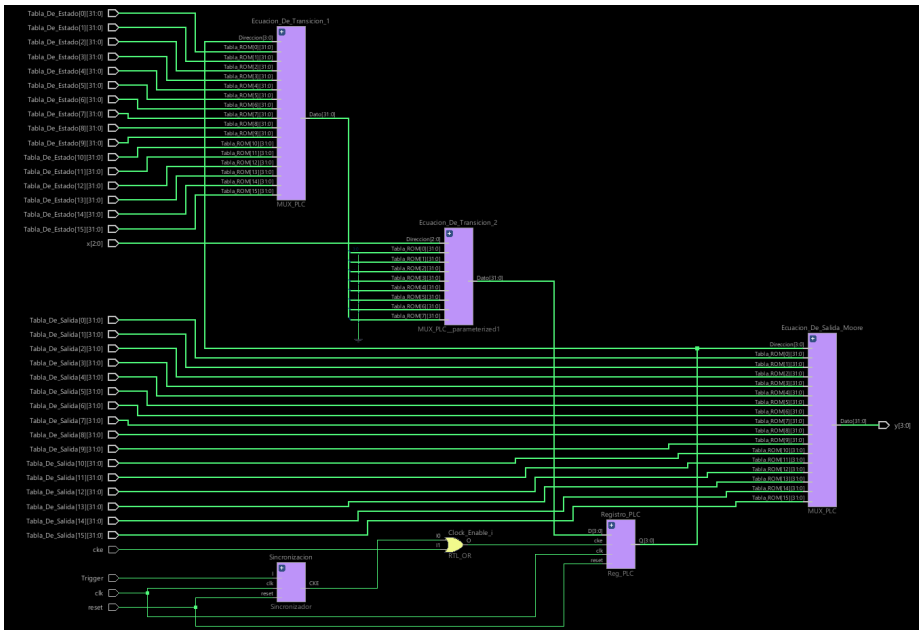

Figura 1: Esquemático RTL de la práctica

En esta sección se va a tratar de resumir la funcionalidad del sistema desde más alto nivel.

Como se puede apreciar en la Figura 2 , los estados se encuentran codificados en la tabla de estados con un tamaño de 16 filas y 32 columnas, que llegan a la ecuación de transición 1, para ser seleccionar los posibles siguientes estados.

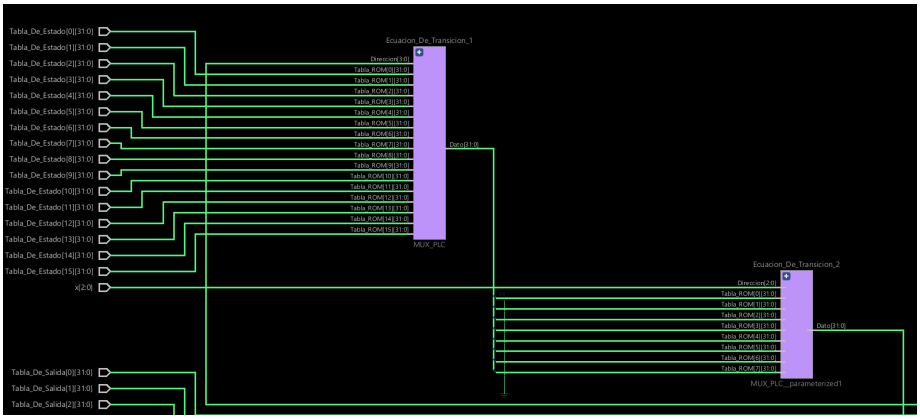

Figura 2: Esquemático RTL de Ecuación de Transición 1 y 2

Posteriormente, en la en la ecuación de transición 2, se selecciona el estado. Al entrar en el registro, se trunca y se mantiene hasta que se active por el reloj o el trigger.

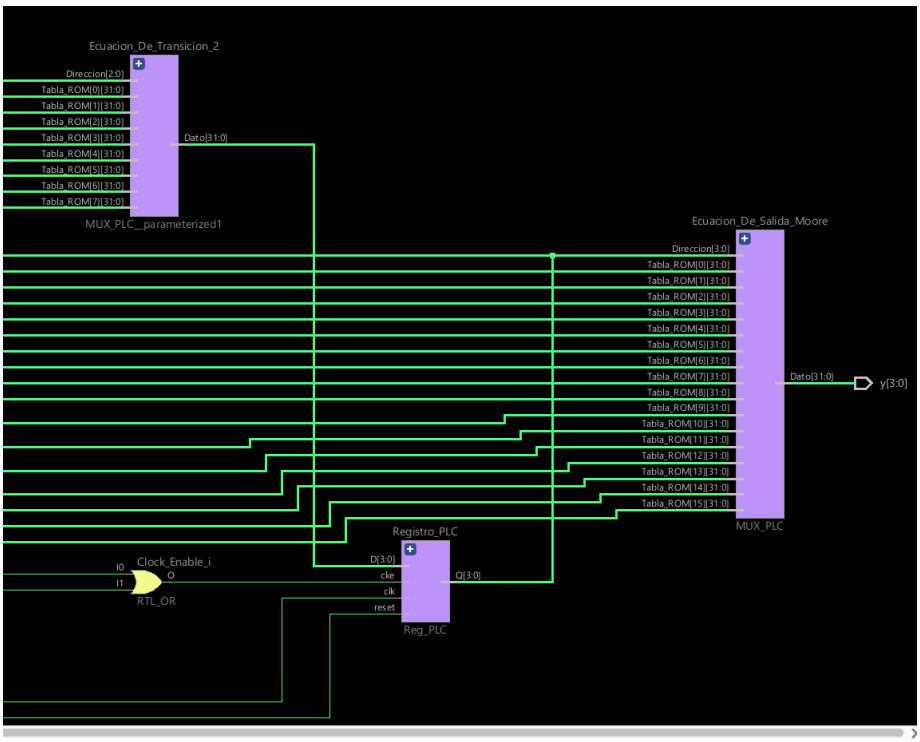

Figura 3: Esquemático RTL de Ecuación de Salida y Registro

Esa dirección, selecciona una salida determinada en la ecuación de salida que tiene como entrada la tabla

de salida. Por otra parte, esa dirección vuelve a entrar en la ecuación de transición 1. La salida del sistema, también finalmente truncada a 4 bits.

3. Diseño del Test Bench

3.1. Diseño de la FSM

Es necesario escribir el código del banco de pruebas de la FSM realizada en la segunda práctica de la asignatura, para ello se realiza un recordatorio del diseño implementado a continuación.

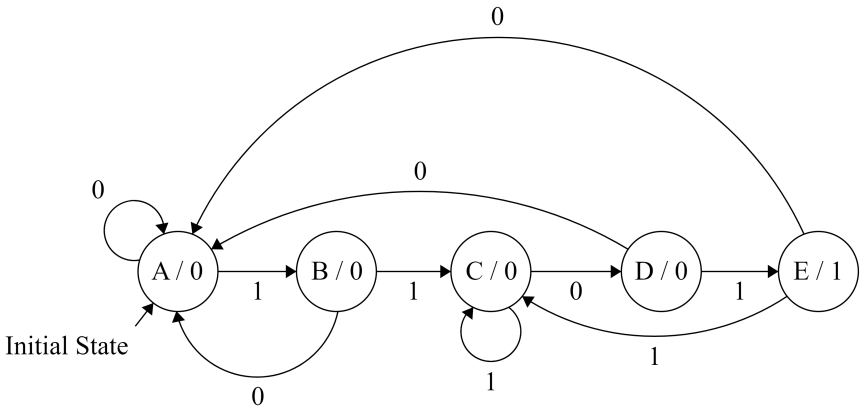

Figura 4: Diagrama de la FSM de Moore

Explicación de los estados:

- **A**: Estado inicial. No hay bit correctos. (Y = 0)
- **B**: Un bit correcto. (Y = 0)
- **C**: Dos bits correctos. (Y = 0)
- **D**: Tres bits correctos. (Y = 0)
- **E**: Cuatro bits y secuencia correcta. (Y = 1)

Cuando se introduce un bit erróneo, se comprueba si la cadena de bits que ha llegado puede ser parte de una nueva secuencia correcta, por lo que no se vuelve directamente al estado inicial.

De esta manera, en los estados **B** y **D** al introducir un **cero**, se vuelve al estado inicial. Sin embargo, en el estado **C**, es posible volver a aprovechar la secuencia de **unos** sucesivos para una nueva secuencia correcta.

Por otra parte, cuando se introduce la secuencia correcta en **E**, es posible usar de nuevo los dos últimos **unos** para la siguiente comprobación de la secuencia, por lo que se vuelve al estado **C**.

3.2. Diseño del código

Se incluyen las tabla de estado y la tabla de salida como comentarios en el Código 1 y en el Código 2, respectivamente, para facilitar su lectura. Posteriormente se codifican en VHDL en sus respectivas señales.

```
1  ----------------------------------
2  -- MOORE
3  ----------------------------------
4  -- Estos estados recogida del ejercicio 2
5  -- Tabla_De_estados_internos
6  -- |    Z(n)   |    X (n)    |
7  -- |           |   1  |   0  |
8  -- | 000 ( A ) | 0001 | 0000 |
9  -- | 001 ( B ) | 0010 | 0000 |
10 -- | 010 ( C ) | 0010 | 0011 |
11 -- | 011 ( D ) | 0100 | 0000 |
12 -- | 100 ( E ) | 0010 | 0000 |
13
14 signal Tabla_De_Estado_interno : Tabla_FSM(0 to 2**m_interno-1) :=
15
16   ((4 => '1', others => '0'),                  -- Estado 0
17    (5 => '1', others => '0'),                  -- Estado 1
18    (5 => '1', 0|1 => '1', others => '0'),      -- Estado 2
19    (6 => '1', others => '0'),                  -- Estado 3
20    (5 => '1', others => '0'),                  -- Estado 4
21    ------------------------------------------------------------
22    (others => '0'),                            -- Estado 5
23    (others => '0'),                            -- Estado 6
24    (others => '0'),                            -- Estado 7
25    (others => '0'),                            -- Estado 8
26    (others => '0'),                            -- Estado 9
27    (others => '0'),                            -- Estado 10
28    (others => '0'),                            -- Estado 11
29    (others => '0'),                            -- Estado 12
30    (others => '0'),                            -- Estado 13
31    (others => '0'),                            -- Estado 14
32    (others => '0')                             -- Estado 15
33    );
```

Código 1: Tabla de estado en Test Bench

El resto del código del banco de pruebas se puede encontrar en Código 3, en el que se realizan la definición de las constantes requeridas, el uso de las señales necesarias para el funcionamiento del sistema además de la configuración necesaria para el uso de ficheros en la lectura de las entradas en el banco de pruebas y la escritura en CSV.

Cabe destacar, la implementación del proceso *trigger* para la simulación los rebotes físicos del conmutador, además de los procesos de *clk*, *cke* y *reset*.

```
1  -- Tabla_De_Salida_interno
2   -- |    Z(n)   | Y(n) |
3   -- | 000 ( A ) |  0   |
4   -- | 001 ( B ) |  0   |
5   -- | 010 ( C ) |  0   |
6   -- | 011 ( D ) |  0   |
7   -- | 100 ( E ) |  1   |
8
9   signal Tabla_De_Salida_interno : Tabla_FSM(0 to 2**m_interno-1) :=
10    ((others => '0'),                           -- Salida: 0
11     (others => '0'),                           -- Salida: 0
12     (others => '0'),                           -- Salida: 0
13     (others => '0'),                           -- Salida: 0
14     (0 => '1', others => '0'),                 -- Salida: 1
15     ------------------------------------------------------------
16     (others => '0'),                           -- Salida: 0
17     (others => '0'),                           -- Salida: 0
18     (others => '0'),                           -- Salida: 0
19     (others => '0'),                           -- Salida: 0
20     (others => '0'),                           -- Salida: 0
```

```
21      (others => '0'),                        -- Salida: 0
22      (others => '0'),                        -- Salida: 0
23      (others => '0'),                        -- Salida: 0
24      (others => '0'),                        -- Salida: 0
25      (others => '0'),                        -- Salida: 0
26      (others => '0')                         -- Salida: 0
27      );
```

Código 2: Tabla de salida en Test Bench

4. Simulación

4.1. Test Bench

Se incluye el archivo completo del banco de pruebas para su consulta en el Código 3.

4.2. Fichero de estímulos Moore

```
#Fichero de Estímulos de práctica 3 Moore
#Device Name: Discovery2NI
#Nombre: Izan Amador, Jorge Benavides
#Fecha: 11 de Enero de 2022.
#
# Delay Time (ns) Input (I).

# Secuencia correcta
    25 ns 001
    25 ns 001
    25 ns 000
    25 ns 001

# Secuencias aleatorias (erróneas)
# Primera secuencia

    25 ns 001
    25 ns 000
    25 ns 001
    25 ns 000
```

Se codifican una secuencia correcta y una secuencia errónea para comprobar el funcionamiento del sistema, y se adaptan los estímulos al tamaño necesario.

4.3. Cronograma de simulación

Se observa en la simulación el correcto funcionamiento de la FSM, ya que se obtiene un uno a la salida cuando se detecta la secuencia correcta, después de pasar por los estados pertinentes.

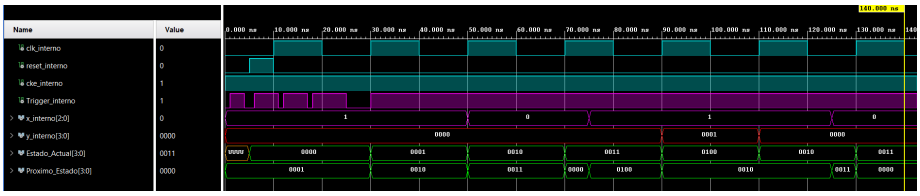

Figura 5: Cronograma de simulación Moore

4.4. Fichero CSV generado

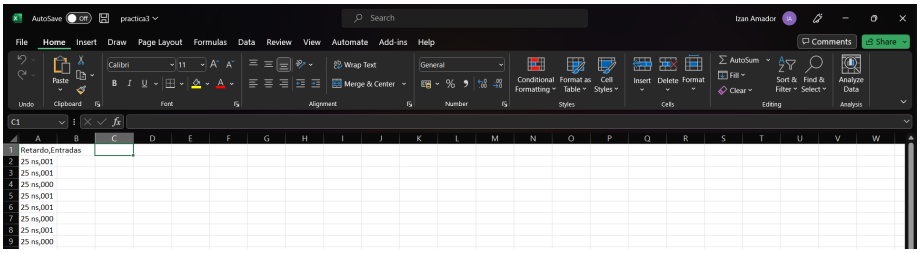

Figura 6: Fichero CSV Generado

Se obtienen los valores esperados correspondientes al fichero de estímulos en el archivo de valores separados mediante comas.

5. Síntesis

Se vuelve a sintetizar el sistema y se comprueba que no existen errores en el archivo de banco de pruebas, tanto de compilación como los añadidos en los procesos de verificación.

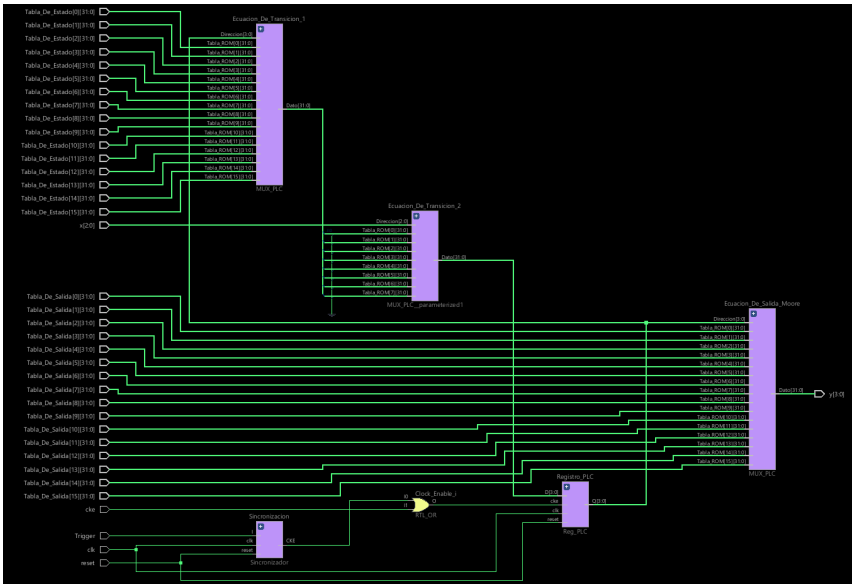

Figura 7: Esquemático RTL de la práctica

6. Anexos

6.1. Test Bench completo

```
--------------------------------------------------------------------------------
-- Company: Universidad de Málaga
-- Engineer: Izan Amador, Jorge L. Benavides
--
-- Create Date: 11-01-2023
-- Design Name: Test_Bench
-- Module Name: Test_Bench - Behavioral
-- Project Name: PLC
-- Target Devices: Zybo
-- Tool Versions: Vivado 2022.1
-- Description: Banco de pruebas para un PLC genérico.
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use STD.textIO.ALL;                     -- Se va a hacer uso de ficheros.
use work.Tipos_FSM_PLC.all;             -- Se importa la librería Tipos_FMS_PLC
                                        -- para trabajar con el tipo Tabla_FSM,
                                        -- el cual es un array de 32 bits

entity Test_Bench is
-- En el banco de pruebas no se definen puertos.
end Test_Bench;

architecture Comportamiento of Test_Bench is
  component FSM_PLC                     -- Se instancia el componente FSM_PLC para realizar el test

    generic(k    : natural := 32;       -- k entradas.
            p    : natural := 32;       -- p salidas.
            m    : natural := 32;       -- m biestables. (Hasta 16 estados)
            T_DM : time    := 10 ps;    -- Tiempo de retardo desde el cambio de dirección del MUX hasta la actualización de la salida Q.
            T_D  : time    := 10 ps;    -- Tiempo de retardo desde el flanco activo del reloj hasta la actualización de la salida Q.
            T_SU : time    := 10 ps;    -- Tiempo de Setup.
            T_H  : time    := 10 ps;    -- Tiempo de Hold.
            T_W  : time    := 10 ps);   -- Anchura de pulso.

    port (x              : in  STD_LOGIC_VECTOR(k - 1 downto 0);  -- x es el bus de entrada.
          y              : out STD_LOGIC_VECTOR(p - 1 downto 0);  -- y es el bus de salida.
          Tabla_De_Estado : in  Tabla_FSM(0 to 2**m - 1);  -- Contiene la Tabla de Estado estilo Moore: Z(n+1)=T1(Z(n),x(n))
          Tabla_De_Salida : in  Tabla_FSM(0 to 2**m - 1);  -- Contiene la Tabla de Salida estilo Moore: Y(n  )=T2(Z(n))
          clk            : in  STD_LOGIC;  -- La señal de reloj.
          cke            : in  STD_LOGIC;  -- La señal de habilitación de avance: si vale '1' el autómata avanza a ritmo de clk y si vale '0' manda Trigger.
          reset          : in  STD_LOGIC;  -- La señal de inicialización.
          Trigger        : in  STD_LOGIC); -- Señal de disparo (single shot) asíncrono y posíblemente con rebotes para hacer un avance único. Lleva un sincronizador.
end component FSM_PLC;


  constant semiperiodo : time    := 10 ns;  -- Constante para el tiempo del reloj
  constant k_interno   : natural := 3;      -- Número de entradas
  constant p_interno   : natural := 4;      -- Número de salidas
  constant m_interno   : natural := 4;      -- Número de biestables (16 estados
                                            -- posibles

  signal x_interno : std_logic_vector(k_interno -1 downto 0) := (others => 'U'); -- Entrada
  signal y_interno : std_logic_vector(p_interno -1 downto 0) := (others => 'U'); -- Salida
  signal reset_interno, cke_interno, Trigger_interno : std_logic := 'U';         -- Señales de control
  signal clk_interno : std_logic := '0';                                         -- Señal del reloj

  -----------------------------------
  -- MOORE
  -----------------------------------
  -- Estos estados recogida del ejercicio 2
  -- Tabla_De_estados_internos
  -- |    Z(n)   |    X (n)    |
```

```
72    -- |           |   1  |   0  |
73    -- | 000 ( A ) | 0001 | 0000 |
74    -- | 001 ( B ) | 0010 | 0000 |
75    -- | 010 ( C ) | 0010 | 0011 |
76    -- | 011 ( D ) | 0100 | 0000 |
77    -- | 100 ( E ) | 0010 | 0000 |
78
79    signal Tabla_De_Estado_interno : Tabla_FSM(0 to 2**m_interno-1) :=
80
81      ((4 => '1', others => '0'),                  -- Estado 0
82       (5 => '1', others => '0'),                  -- Estado 1
83       (5 => '1', 0|1 => '1', others => '0'),      -- Estado 2
84       (6 => '1', others => '0'),                  -- Estado 3
85       (5 => '1', others => '0'),                  -- Estado 4
86       ------------------------------------------------------------
87       (others => '0'),                            -- Estado 5
88       (others => '0'),                            -- Estado 6
89       (others => '0'),                            -- Estado 7
90       (others => '0'),                            -- Estado 8
91       (others => '0'),                            -- Estado 9
92       (others => '0'),                            -- Estado 10
93       (others => '0'),                            -- Estado 11
94       (others => '0'),                            -- Estado 12
95       (others => '0'),                            -- Estado 13
96       (others => '0'),                            -- Estado 14
97       (others => '0')                             -- Estado 15
98       );
99
100   -- Tabla_De_Salida_interno
101   -- |    Z(n)   | Y(n) |
102   -- | 000 ( A ) |  0   |
103   -- | 001 ( B ) |  0   |
104   -- | 010 ( C ) |  0   |
105   -- | 011 ( D ) |  0   |
106   -- | 100 ( E ) |  1   |
107
108   signal Tabla_De_Salida_interno : Tabla_FSM(0 to 2**m_interno-1) :=
109     ((others => '0'),                             -- Salida: 0
110      (others => '0'),                             -- Salida: 0
111      (others => '0'),                             -- Salida: 0
112      (others => '0'),                             -- Salida: 0
113      (0 => '1', others => '0'),                   -- Salida: 1
114      ------------------------------------------------------------
115      (others => '0'),                             -- Salida: 0
116      (others => '0'),                             -- Salida: 0
117      (others => '0'),                             -- Salida: 0
118      (others => '0'),                             -- Salida: 0
119      (others => '0'),                             -- Salida: 0
120      (others => '0'),                             -- Salida: 0
121      (others => '0'),                             -- Salida: 0
122      (others => '0'),                             -- Salida: 0
123      (others => '0'),                             -- Salida: 0
124      (others => '0'),                             -- Salida: 0
125      (others => '0')                              -- Salida: 0
126      );
127
128   begin
129
130     DUT : FSM_PLC     -- Device Under Test
131       generic map (   -- Asociación de constantes definidas en la arquitectura
132         k => k_interno,
133         p => p_interno,
134         m => m_interno)
135       port map(         -- Asociación de los puertos con sus respectivas señales
136         -- para simulación
137         x => x_interno,
138         y => y_interno,
139         Tabla_De_Estado => Tabla_De_Estado_interno,
140         Tabla_De_Salida => Tabla_De_Salida_interno,
141         Trigger => Trigger_interno,
142         reset   => reset_interno,
143         clk     => clk_interno,
144         cke     => cke_interno);
145
146     -- Extraido de The Student Guide to VHDL, Peter J.Asheden
147     clock_gen : process (clk_interno) is -- Reloj del sistema
148     begin
149       if clk_interno = '0' then
150         clk_interno <= '1' after semiperiodo,
```

```
                    '0' after 2*semiperiodo;
    end if;
  end process clock_gen;

  cke_interno <= '1'; -- Habilitación del PLC

  trigger: process    -- Simulación de rebotes como si fuese un conmutador externo
  begin
    trigger_interno <= '0';
    wait for 1 ns;
    trigger_interno <= '1';
    wait for 3 ns;
    trigger_interno <= '0';
    wait for 2 ns;
    trigger_interno <= '1';
    wait for 5 ns;
    trigger_interno <= '0';
    wait for 1 ns;
    trigger_interno <= '1';
    wait for 5 ns;
    trigger_interno <= '0';
    wait for 1 ns;
    trigger_interno <= '1';
    wait for 7 ns;
    trigger_interno <= '0';
    wait for 5 ns;
    trigger_interno <= '1';
    wait;
  end process trigger;

  reset : process         -- Señal de reset
  begin
    reset_interno <= '0';
    wait for 5 ns;
    reset_interno <= '1';
    wait for 5 ns;
    reset_interno <= '0';
    wait;
  end process reset;

  Estimulos_Desde_Fichero : process

    file Input_File  : text;
    file Output_File : text;

    variable Input_Data   : BIT_VECTOR(k_interno-1 downto 0) := (OTHERS => '0');    -- La entrada del fichero de estímulos está condicionada al tamaño de la
                                                                                    -- entrada del PLC, que en este caso es de k_interno
    variable Delay        : time                     := 0 ms;
    variable Input_Line   : line                     := NULL;
    variable Output_Line  : line                     := NULL;
    variable Std_Out_Line : line                     := NULL;
    variable Correcto     : Boolean                  := True;
    constant Coma         : character                := ',';

  begin
-- practica3_Estimulos.txt contiene los estímulos y los tiempos de retardo para el semisumador.
    file_open(Input_File, "C:\Users\izana\Documents\GitHub\SEA\Estimulos\practica3_mealy_Estimulos.txt", read_mode);
    --file_open(Input_File, "C:\\Users\\izana\\Documents\GitHub\SEA\Estimulos\practica3_Estimulos.txt", read_mode);

-- practica_Estimulos.csv contiene los estímulos y los tiempos de retardo para el Analog Discovery 2.
    file_open(Output_File, "C:\Users\izana\Documents\GitHub\SEA\CSV\practica3_mealy.csv", write_mode);
    -- file_open(Output_File, "C:\Users\izana\Documents\GitHub\SEA\CSV\practica3.csv", write_mode);

-- Titles: Son para el formato EXCEL *.CSV (Comma Separated Values):
    write(Std_Out_Line, string'("Retardo"), right, 7);
    write(Std_Out_Line, Coma, right, 1);
    write(Std_Out_Line, string'("Entradas"), right, 8);

    Output_Line := Std_Out_Line;

    writeline(output, Std_Out_Line);
    writeline(Output_File, Output_Line);

    while (not endfile(Input_File)) loop

      readline(Input_File, Input_Line);

      read(Input_Line, Delay, Correcto);  -- Comprobación de que se trata de un texto que representa
      -- el retardo, si no es así leemos la siguiente línea.
```

```
230       if Correcto then
231
232         read(Input_Line, Input_Data);  -- El siguiente campo es el vector de pruebas.
233         x_interno <= TO_STDLOGICVECTOR(Input_Data);
234         -- De forma simultánea lo volcaremos en consola en csv.
235         write(Std_Out_Line, Delay, right, 5);  -- Longitud del retardo, ej. "20 ms".
236         write(Std_Out_Line, Coma, right, 1);
237         write(Std_Out_Line, Input_Data, right, 2);  --Longitud de los datos de entrada.
238
239         Output_Line := Std_Out_Line;
240
241         writeline(output, Std_Out_Line);
242         writeline(Output_File, Output_Line);
243
244         wait for Delay;
245       end if;
246     end loop;
247
248     file_close(Input_File);             -- Cerramos el fichero de entrada.
249     file_close(Output_File);            -- Cerramos el fichero de salida.
250     wait;
251   end process Estimulos_Desde_Fichero;
252
253
254 end Comportamiento;
```

Código 3: Test Bench Completo