



UNIVERSIDAD  
DE MÁLAGA

| **uma.es**

José Luis Cambil Calderón  
Juan María Herrera López

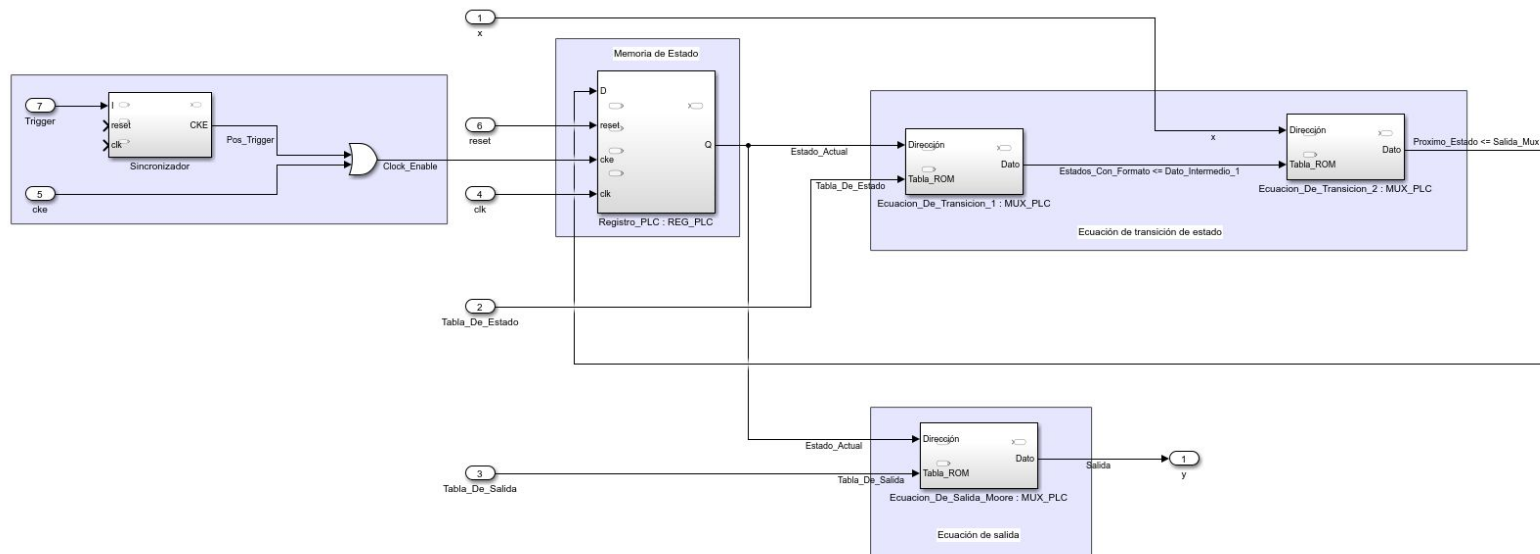
# SEA - Trabajo final - PLC

PLC de Juan María Herrera López - Control de nivel de un depósito

## Índice

- Estudio previo del código
  - Estructura general
  - Comprobación de  $k$ ,  $m$ ,  $p$ ,  $T_{SU}$ ,  $T_H$ ,  $T_W$
  - Sincronización - Clock\_Enable
  - Registro de Estado
  - Ecuación de Transición de Estado
  - Ecuación de Salida
- Simulación
  - FSM implementada
  - Código VHDL del Test Bench
  - Fichero de estímulos
  - Resultado
- Síntesis RTL

## Estudio previo - Estructura general

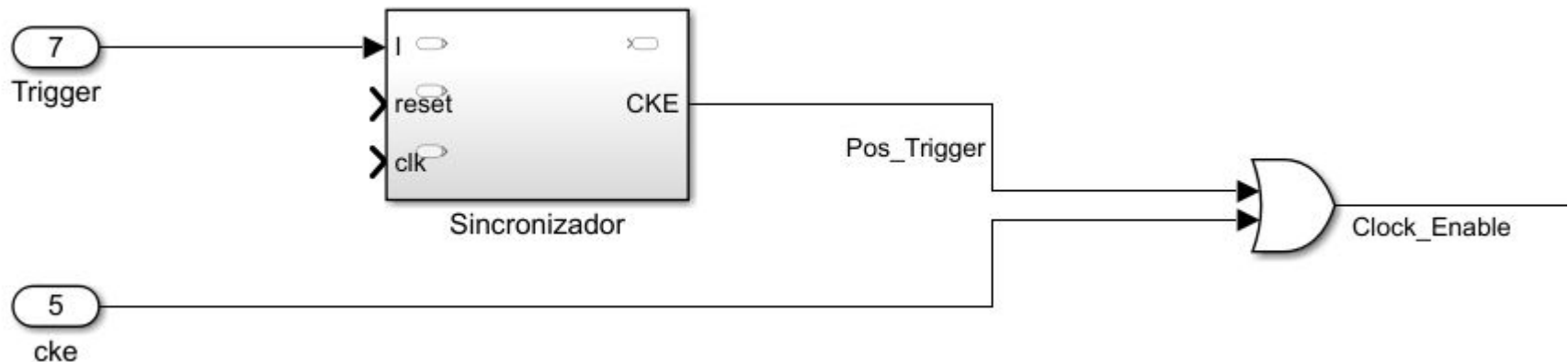


- 4 bloques
  - Sincronizador
  - Registro
  - Ecuación de transición de estado
  - Ecuación de salida

## Estudio previo - Comprobación de parámetros

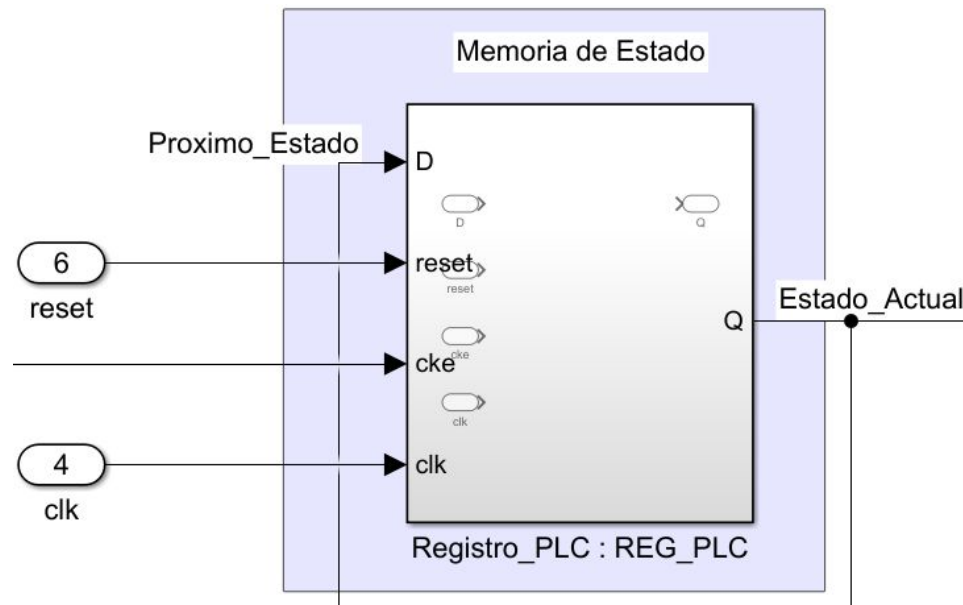
- Comprueba  $2^k \cdot m \leq 32$
- Comprueba  $2^k \cdot p \leq 32$
- Parámetros de tamaño aceptables
- Comprueba validez de clk
  - T set-up
  - T hold
  - T width

## Estudio previo - Sincronización de Clock Enable



- Permite usar tanto Trigger como cke
- Clock\_Enable va al registro

## Estudio previo - Registro de Estado

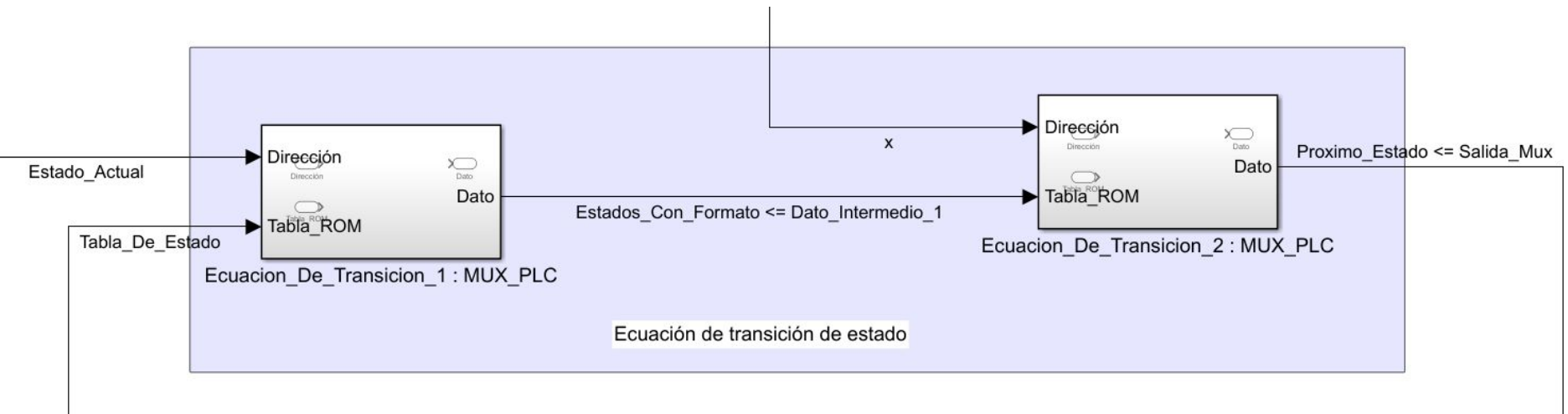


- Implementa la memoria de estado, FF-D
- $\text{cke} \leq \text{Clock\_Enable}$

## Estudio previo - Ecuación de Transición de Estado

- Implementada con tres procesos principalmente
  - MUX1 - Busca en Tabla\_De\_Entrada en función de Estado\_Actual
  - Asignacion\_MUX - Formate el vector de salida de MUX1 en otra tabla, Estados\_Con\_Formato
  - MUX2 - Busca en Estados\_Con\_Formato el Estado\_Siguiente en función de la entrada x.

## Estudio previo - Ecuación de Transición de Estado



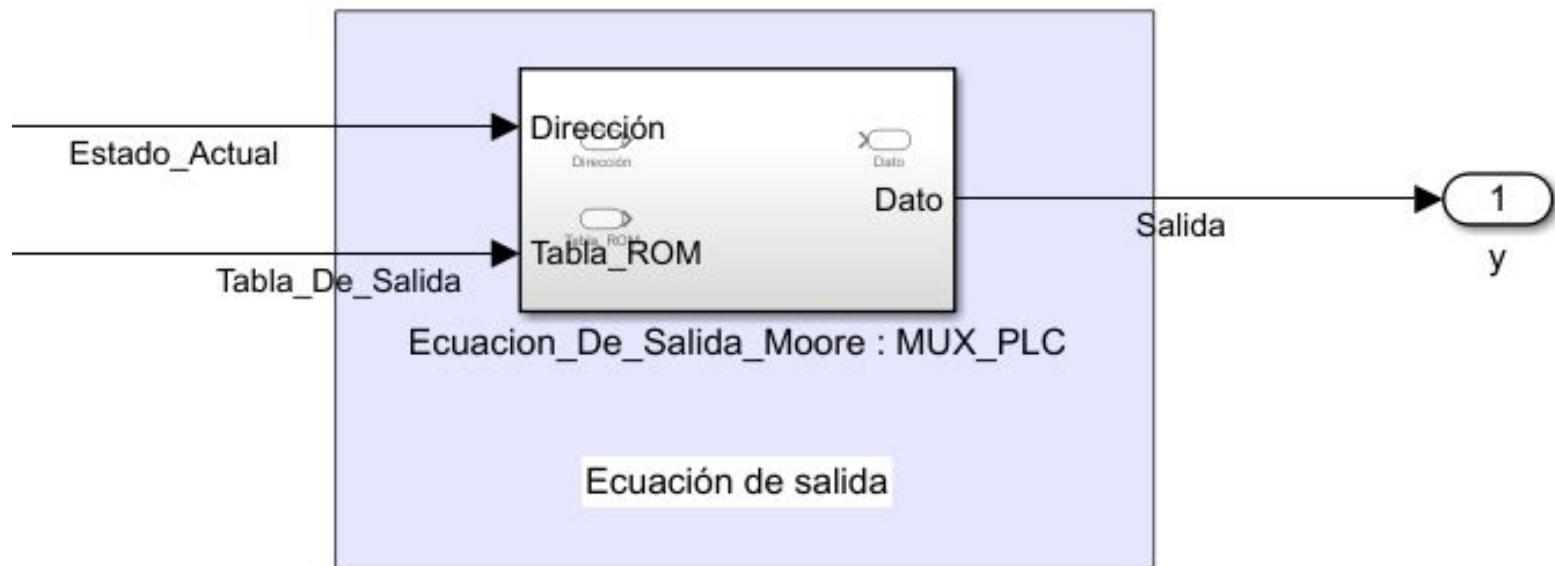


# Estudio previo - Ecuación de Transición de Estado

## - Formato Tabla\_De\_Estado

```
constant Tabla_De_Estado : Tabla_FSM := (  
  -- Entrada:  
  --      7      6      5      4      3      2      1      0  
  --  
  -- Estado:  
  b"0000_0000_0000_0000_0011_0010_0001_0000", -- 0  B  
  b"0000_0000_0000_0000_0011_0010_0001_0000", -- 1  MBS  
  b"0000_0000_0000_0000_0011_0010_0001_0000", -- 2  MAS  
  b"0000_0000_0000_0000_0011_0100_0101_0000", -- 3  A  
  b"0000_0000_0000_0000_0011_0100_0101_0000", -- 4  MAB  
  b"0000_0000_0000_0000_0011_0100_0101_0000", -- 5  MBB  
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 6  
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 7  
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 8  
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 9  
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 10  
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 11  
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 12  
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 13  
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 14  
  b"0000_0000_0000_0000_0000_0000_0000_0000" -- 15
```

## Estudio previo - Ecuación de Salida



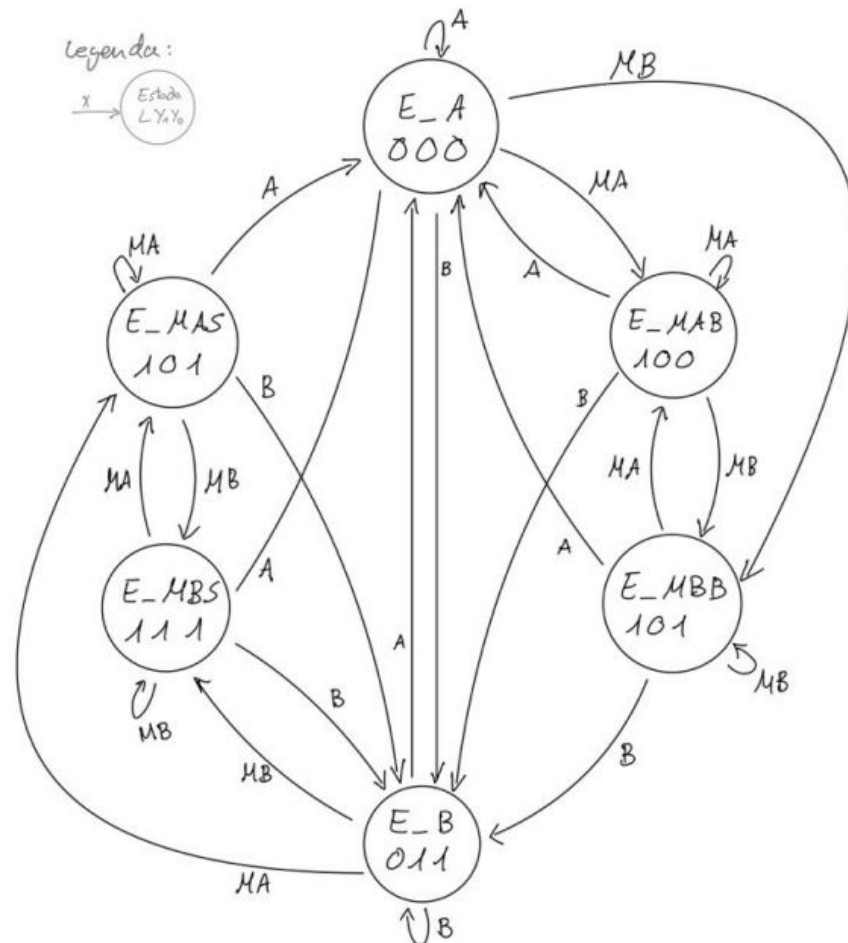
## Estudio previo - Ecuación de Salida - Formato Tabla\_De\_Salida

```
constant Tabla_De_Salida : Tabla_FSM := (  
  --                                Estado:  
  x"00000003", -- 0    B  
  x"00000007", -- 1    MBS  
  x"00000005", -- 2    MAS  
  x"00000000", -- 3    A  
  x"00000004", -- 4    MAB  
  x"00000005", -- 5    MBB  
  x"00000000", -- 6  
  x"00000000", -- 7  
  x"00000000", -- 8  
  x"00000000", -- 9  
  x"00000000", -- 10  
  x"00000000", -- 11  
  x"00000000", -- 12  
  x"00000000", -- 13  
  x"00000000", -- 14  
  x"00000000"  -- 15  
);
```

## Simulación - FSM implementada

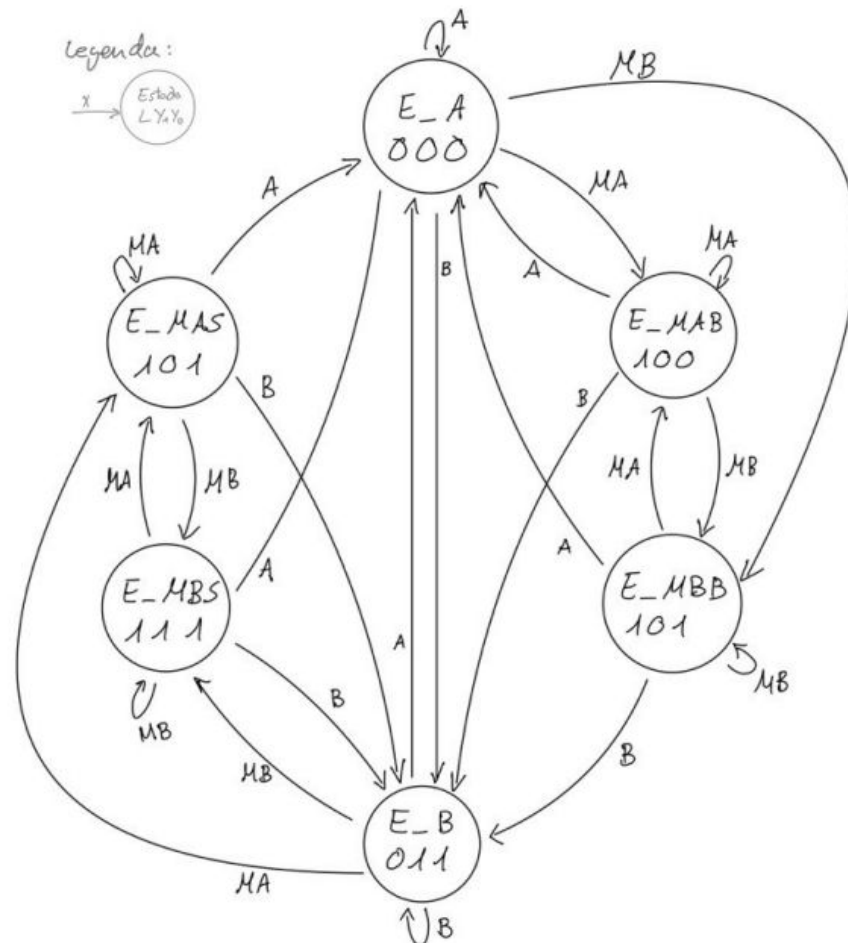
### Control de nivel de un depósito

- $k = 2$  bits entrada
- $p = 3$  bits salida
- $m = 3$  bits estado



## Simulación - FSM implementada

- B - BAJO - 0
- MBS - MEDIO BAJO SUBIENDO - 1
- MAS - MEDIO ALTO SUBIENDO - 2
- A - ALTO - 3
- MAB - MEDIO ALTO BAJANDO - 4
- MBB - MEDIO BAJO BAJANDO - 5



## Simulación - Código VHDL del Test Bench

```
component FSM_PLC is
  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 MU
           T_D    : time    := 10 ps; -- Tiempo de retardo desde el flanco activo del reloj ha
           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
        Tabla_De_Salida : in Tabla_FSM( 0 to 2**m - 1 ); -- Contiene la Tabla de Salida
        clk      : in STD_LOGIC;    -- La señal de reloj.
        cke      : in STD_LOGIC;    -- La señal de habilitación de avance: si vale '1' el
        reset    : in STD_LOGIC;    -- La señal de inicialización.
        Trigger  : in STD_LOGIC );  -- La señal de disparo (single shot) asíncrono y posib
end component;
```



## Simulación - Código VHDL del Test Bench

```
constant k : natural := k_Max;  -- entradas
constant p : natural := p_Max;  -- salidas
constant m : natural := m_Max;  -- biestables

constant Medio_Periodo : Time := 5 ns;

signal x : std_logic_vector(k-1 downto 0) := (others=>'0');
signal y : std_logic_vector(p-1 downto 0) := (others=>'0');
signal clk      : std_logic := '0';
signal cke      : std_logic := '0';
signal reset    : std_logic := '0';
signal Trigger  : std_logic := '0';
```

## Simulación - Código VHDL del Test Bench

```
-- Ecuación de transición de estado
constant Tabla_De_Estado : Tabla_FSM := (
  -- Entrada:
  --      7      6      5      4      3      2      1      0
  --
  -- Estado:
  b"0000_0000_0000_0000_0011_0010_0001_0000", -- 0  B
  b"0000_0000_0000_0000_0011_0010_0001_0000", -- 1  MBS
  b"0000_0000_0000_0000_0011_0010_0001_0000", -- 2  MAS
  b"0000_0000_0000_0000_0011_0100_0101_0000", -- 3  A
  b"0000_0000_0000_0000_0011_0100_0101_0000", -- 4  MAB
  b"0000_0000_0000_0000_0011_0100_0101_0000", -- 5  MBB
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 6
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 7
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 8
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 9
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 10
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 11
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 12
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 13
  b"0000_0000_0000_0000_0000_0000_0000_0000", -- 14
  b"0000_0000_0000_0000_0000_0000_0000_0000" -- 15
);
```



## Simulación - Código VHDL del Test Bench

```
constant Tabla_De_Salida : Tabla_FSM := (  
    --                                Estado:  
    x"00000003", -- 0    B  
    x"00000007", -- 1    MBS  
    x"00000005", -- 2    MAS  
    x"00000000", -- 3    A  
    x"00000004", -- 4    MAB  
    x"00000005", -- 5    MBB  
    x"00000000", -- 6  
    x"00000000", -- 7  
    x"00000000", -- 8  
    x"00000000", -- 9  
    x"00000000", -- 10  
    x"00000000", -- 11  
    x"00000000", -- 12  
    x"00000000", -- 13  
    x"00000000", -- 14  
    x"00000000"  -- 15  
);
```

## Simulación - Código VHDL del Test Bench

```
begin

    DUT : FSM_PLC
        generic map (k=>k, p=>p, m=>m)
        port map (x=>x,y=>y,Tabla_De_Estado=>Tabla_De_Estado,
        Tabla_De_Salida=>Tabla_De_Salida,clk=>clk,cke=>cke,
        reset=>reset,Trigger=>Trigger);

    Reloj : process
    begin
        clk <= '0';
        wait for Medio_Periodo;
        clk <= '1';
        wait for Medio_Periodo;
    end process Reloj;

    Inicializacion : process
    begin
        reset<='1';
        wait for 3ns;
        reset<='0';
        wait;
    end process;
```

## Simulación - Código VHDL del Test Bench

```
Rebotes : process
begin
    Trigger <= '0';
    wait for 20 ns;
    for i in 0 to 4 loop
        Trigger <= '0';
        wait for 7870ps;
        Trigger <= '1';
        wait for 5210ps;
    end loop;
    wait for 200ns;
    for i in 0 to 7 loop
        Trigger <= '0';
        wait for 2000ps;
        Trigger <= '1';
        wait for 967ps;
    end loop;
    Trigger <= '0';
    wait for (780ns - 89136 ps);
end process Rebotes;
```

## Simulación - Código VHDL del Test Bench

```
Estimulos_Desde_Fichero : process

    file Input_File : text;
    file Output_File : text;

    variable      Input_Data : BIT_VECTOR(k-1 downto 0 ) := ( OTHERS => '0' );
    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

    -- estimulos.txt contiene los estímulos y los tiempos de retardo.
    file_open( Input_File, "C:\Users\Usuario\Desktop\estimulos.txt", read_mode );
    -- etimulos.csv contiene los estímulos y los tiempos de retardo para el Analog Discovery 2.
    file_open( Output_File, "C:\Users\Usuario\Desktop\estimulos.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;
```

## Simulación - Código VHDL del Test Bench

```
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.

    if Correcto then

        read( Input_Line, Input_Data );    -- El siguiente campo es el vector de pruebas.

        x <= TO_STDLOGICVECTOR( Input_Data )(k-1 downto 0);
                                           -- De forma simultánea lo volcaremos en consola en csv.

        write( Std_Out_Line,      Delay, right, 5 ); -- Longitud del retardo, ej. "20 ms".
        write( Std_Out_Line,      Coma, right, 1 );
        write( Std_Out_Line,      Input_Data, right, 2 ); --Longitud de los datos de entrada.

        Output_Line := Std_Out_Line;

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

        wait for Delay;
    end if;
end loop;

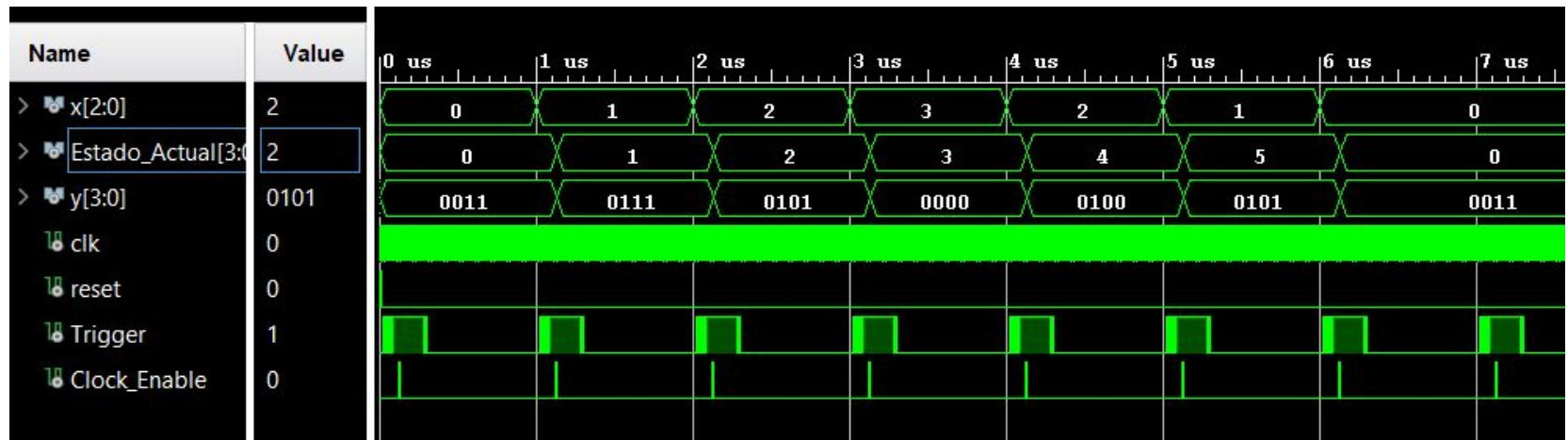
file_close( Input_File ); -- Cerramos el fichero de entrada.
file_close( Output_File ); -- Cerramos el fichero de salida.
wait;
end process Estimulos_Desde_Fichero;
```

# Simulación - Fichero de Estímulos

Delay Time (ns)	Input (x).
1000 ns	000
1000 ns	001
1000 ns	010
1000 ns	011
1000 ns	010
1000 ns	001
1000 ns	000
1000 ns	000
1000 ns	001
1000 ns	001
1000 ns	010
1000 ns	010
1000 ns	011
1000 ns	011
1000 ns	010
1000 ns	010
1000 ns	001
1000 ns	001
1000 ns	000
1000 ns	010
1000 ns	011
1000 ns	001
1000 ns	010
1000 ns	000

	A	
1	Retardo,Entradas	
2	1000 ns,000	
3	1000 ns,001	
4	1000 ns,010	
5	1000 ns,011	
6	1000 ns,010	
7	1000 ns,001	
8	1000 ns,000	
9	1000 ns,000	
10	1000 ns,001	
11	1000 ns,001	
12	1000 ns,010	
13	1000 ns,010	
14	1000 ns,011	
15	1000 ns,011	
16	1000 ns,010	
17	1000 ns,010	
18	1000 ns,001	
19	1000 ns,001	
20	1000 ns,000	
21	1000 ns,010	
22	1000 ns,011	
23	1000 ns,001	
24	1000 ns,010	
25	1000 ns,000	
26		

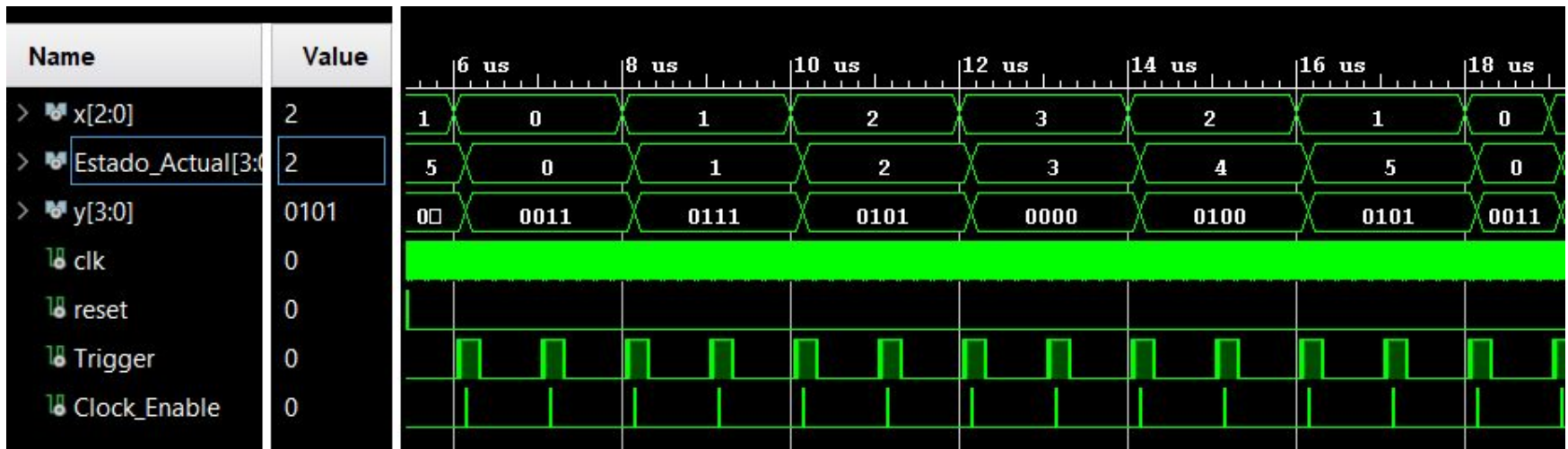
## Simulación - Resultado



Ciclo de subida y bajada rápido



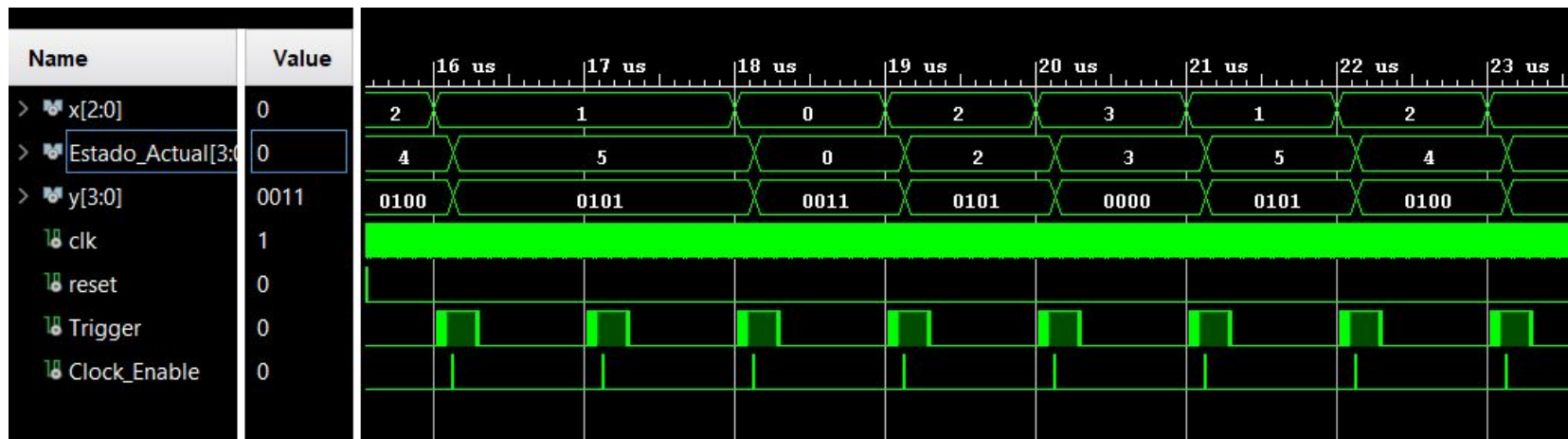
## Simulación - Resultado



Ciclo de subida y bajada lento



## Simulación - Resultado



Saltos entre estados

# Síntesis RTL

