Instituto Politécnico Nacional

Escuela Superior de Cómputo

Diseño de Sistemas Digitales

Práctica 13: Marquesina

Integrantes: Bravo Esquivel Gustavo

Colín Ramiro Joel

Pasten Juarez Joshua Michael

Profesor: Mújica Ascencio Cesar

Grupo: 4CV3

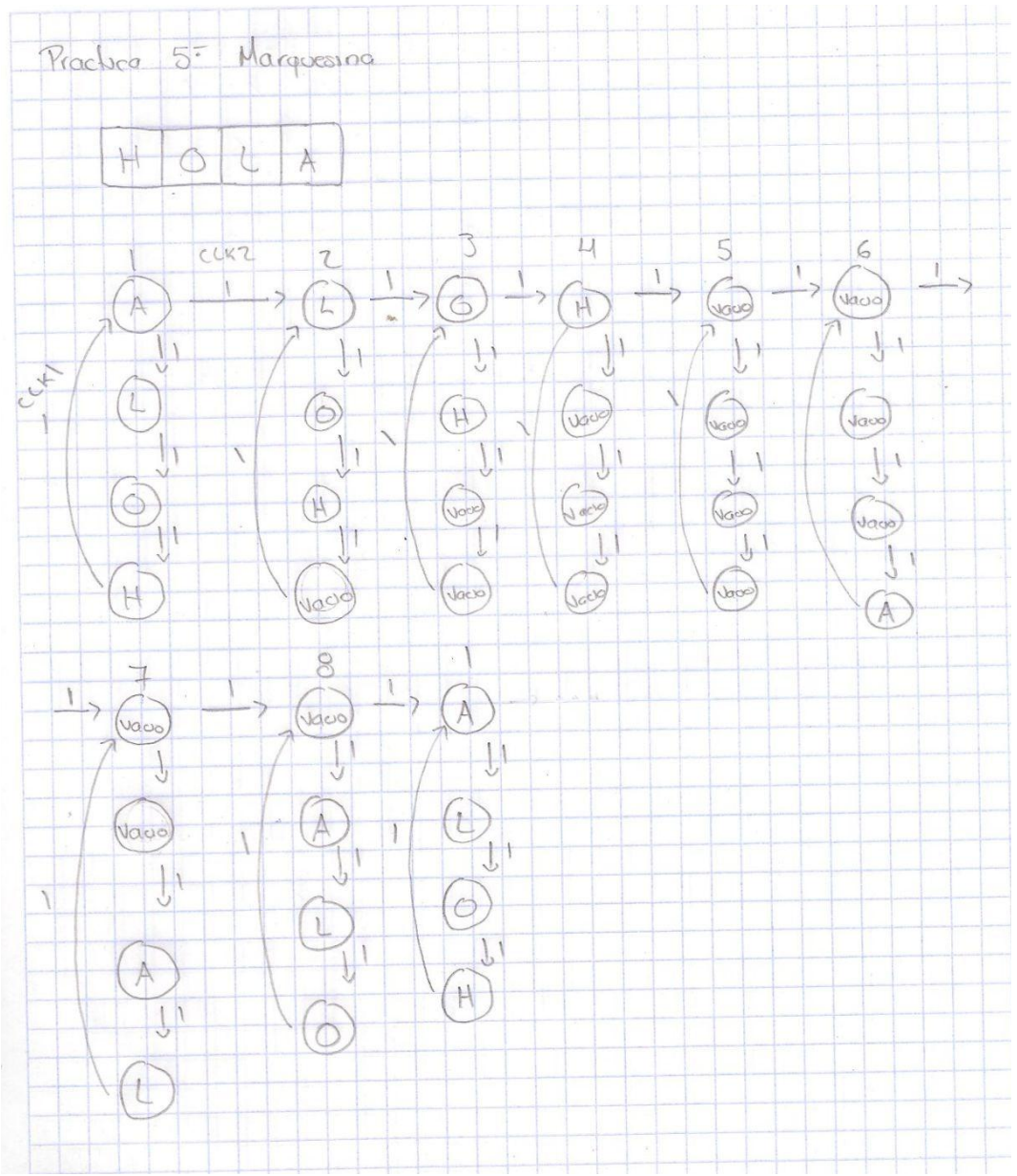
I. Introducción

Introducción

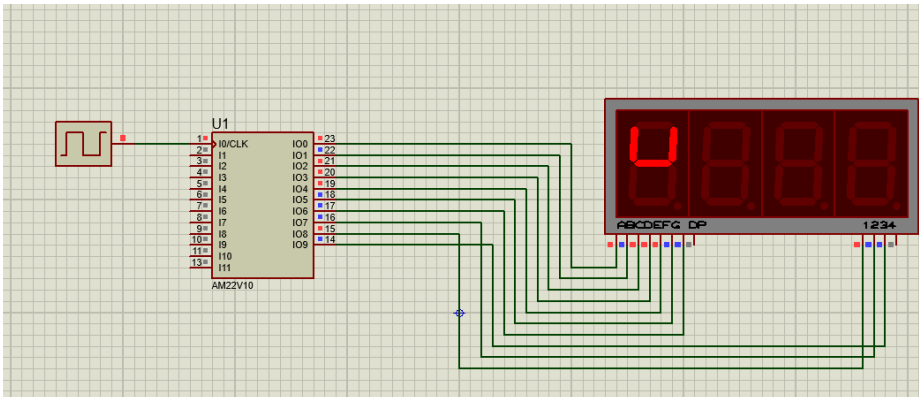
Una Marquesina es un tipo de letrero que usa luces led para mostrar un mensaje, sin embargo este mensaje no es fijo, sino que cierto tiempo va recorriendo dentro del lugar que se muestra dicho mensaje, y, a su vez, se recorre, desapareciendo así de la visibilidad de las personas.

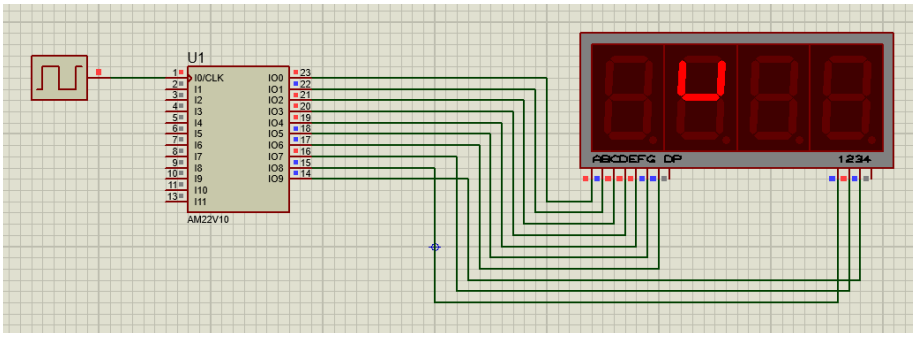
En esta práctica realicamos una especie de marquesina en VHDL gracias a las máquinas de estados. Esta marquesina o mejor dicho, el display, desplegará un mensaje que se irá recorriendo de un lado al otro, replicando una marquesina real.

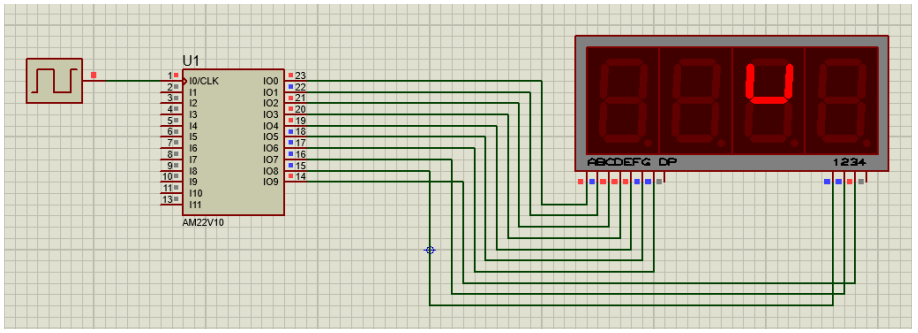
II. Desarrollo



III. Simulaciones







IV. Código VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;
use ieee.numeric_std.all;

entity marquesina is port(
		clk: in std_logic;
		display :out std_logic_vector(6 downto 0);
		ctrl: out std_logic_vector(2 downto 0));
		attribute pin_numbers of marquesina: entity is "clk:1 display(6):23 display(5):22 display(4):21 display(3):20
display(2):19 display(1):18 display(0):17 ctrl(1):16 ctrl(0):15 ";
		end entity;


		architecture relojlento of marquesina is
		--type estados is(L1);
		constant  LH: std_logic_vector (6 downto 0):= "1011100";
		--constant  LI: std_logic_vector (6 downto 0):= "1001111";
		--constant  D1: std_logic_vector (1 downto 0):= "01";
		--constant  D2: std_logic_vector (1 downto 0):= "11";
```

```
--constant  D3: std_logic_vector (1 downto 0):= "10";
type max_estados is(E1,E2,E3);
signal x:max_estados;
--signal conteo2: std_logic_vector(1 downto 0);

begin
process(clk)
begin
if(clk'event and clk='1')then
        case x is
        when  E1=>
                ctrl<="001";
                display<=LH;
                x<=E2;
        when  E2=>
                ctrl<="010";
                display<=LH;
                x<=E3;
        when  E3=>
                ctrl<="100";
                display<=LH;
                x<=E1;
        end case;
        end if;
end process;
end architecture;
```

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

entity Marquesina is
  Port ( clk : in  STD_LOGIC;
       clr : in  STD_LOGIC;
       anodos_7seg : out  STD_LOGIC_VECTOR (3 downto 0);
       display_7seg : out  STD_LOGIC_VECTOR (6 downto 0);
       --contador : inout  STD_LOGIC_VECTOR (3 downto 0);
       en : in  STD_LOGIC);
end Marquesina;

architecture Marquesina of Marquesina is

        type  MACRO_ESTADOS is (A,B,C,D,E,F,G,H);
        signal  MACRO_EDO_ACT,  MACRO_EDO_SIG:  MACRO_ESTADOS;

        type  MICRO_ESTADOS is (MA,MB,MC,MD);
        signal  MICRO_EDO_ACT,  MICRO_EDO_SIG:  MICRO_ESTADOS;

        constant  D0: std_logic_vector (3 downto 0):="1110";
        constant  D1: std_logic_vector (3 downto 0):="1101";
        constant  D2: std_logic_vector (3 downto 0):="1011";
        constant  D3: std_logic_vector (3 downto 0):="0111";
```

```
	constant LETRA_VACIO: std_logic_vector (6 downto 0):="1111111";
	constant LETRA_h: std_logic_vector (6 downto 0):="0001001";
	constant LETRA_O: std_logic_vector (6 downto 0):="1000000";
	constant LETRA_L: std_logic_vector (6 downto 0):="1000111";
	constant LETRA_A: std_logic_vector (6 downto 0):="0001000";

	signal cont : integer:=0;
	signal cont2: integer:=0;
begin
	PROCESS (MACRO_EDO_ACT) BEGIN
		CASE MACRO_EDO_ACT IS
			WHEN A =>
				MACRO_EDO_SIG<= B;
			WHEN B =>
				MACRO_EDO_SIG<= C;
			WHEN C =>
				MACRO_EDO_SIG<= D;
			WHEN D =>
				MACRO_EDO_SIG<= E;
			WHEN E =>
				MACRO_EDO_SIG<= F;
			WHEN F =>
				MACRO_EDO_SIG<= G;
			WHEN G =>
				MACRO_EDO_SIG<= H;
			WHEN H =>
				MACRO_EDO_SIG<= A;
		END CASE;
	END PROCESS;

	PROCESS (MICRO_EDO_ACT,MACRO_EDO_ACT) BEGIN
		CASE MACRO_EDO_ACT IS
			WHEN A =>
				CASE MICRO_EDO_ACT IS
					WHEN MA =>
						display_7seg <= LETRA_A;
						anodos_7seg<=D0;
						MICRO_EDO_SIG<= MB;
					WHEN MB =>
						display_7seg <= LETRA_L;
						anodos_7seg<=D1;
						MICRO_EDO_SIG<= MC;
					WHEN MC =>
						display_7seg <= LETRA_O;
						anodos_7seg<=D2;
```

```
                MICRO_EDO_SIG<= MD;
            WHEN MD =>
                display_7seg <= LETRA_h;
                anodos_7seg<=D3;
                MICRO_EDO_SIG<= MA;
        END CASE;
WHEN B =>
        CASE MICRO_EDO_ACT IS
            WHEN MA =>
                display_7seg <= LETRA_L;
                anodos_7seg<=D0;
                MICRO_EDO_SIG<= MB;
            WHEN MB =>
                display_7seg <= LETRA_O;
                anodos_7seg<=D1;
                MICRO_EDO_SIG<= MC;
            WHEN MC =>
                display_7seg <= LETRA_h;
                anodos_7seg<=D2;
                MICRO_EDO_SIG<= MD;
            WHEN MD =>
                display_7seg <= LETRA_VACIO;
                anodos_7seg<=D3;
                MICRO_EDO_SIG<= MA;
        END CASE;
WHEN C =>
        CASE MICRO_EDO_ACT IS
            WHEN MA =>
                display_7seg <= LETRA_O;
                anodos_7seg<=D0;
                MICRO_EDO_SIG<= MB;
            WHEN MB =>
                display_7seg <= LETRA_h;
                anodos_7seg<=D1;
                MICRO_EDO_SIG<= MC;
            WHEN MC =>
                display_7seg <= LETRA_VACIO;
                anodos_7seg<=D2;
                MICRO_EDO_SIG<= MD;
            WHEN MD =>
                display_7seg <= LETRA_VACIO;
                anodos_7seg<=D3;
                MICRO_EDO_SIG<= MA;
        END CASE;
WHEN D =>
        CASE MICRO_EDO_ACT IS
```

```
                WHEN MA =>
                        display_7seg <= LETRA_A;
                        anodos_7seg<=D0;
                        MICRO_EDO_SIG<= MB;
                WHEN MB =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D1;
                        MICRO_EDO_SIG<= MC;
                WHEN MC =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D2;
                        MICRO_EDO_SIG<= MD;
                WHEN MD =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D3;
                        MICRO_EDO_SIG<= MA;
        END CASE;
WHEN E =>
        CASE MICRO_EDO_ACT IS
                WHEN MA =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D0;
                        MICRO_EDO_SIG<= MB;
                WHEN MB =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D1;
                        MICRO_EDO_SIG<= MC;
                WHEN MC =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D2;
                        MICRO_EDO_SIG<= MD;
                WHEN MD =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D3;
                        MICRO_EDO_SIG<= MA;
        END CASE;
WHEN F =>
        CASE MICRO_EDO_ACT IS
                WHEN MA =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D0;
                        MICRO_EDO_SIG<= MB;
                WHEN MB =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D1;
                        MICRO_EDO_SIG<= MC;
```

```
                WHEN MC =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D2;
                        MICRO_EDO_SIG<= MD;
                WHEN MD =>
                        display_7seg <= LETRA_A;
                        anodos_7seg<=D3;
                        MICRO_EDO_SIG<= MA;
        END CASE;
WHEN G =>
        CASE MICRO_EDO_ACT IS
                WHEN MA =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D0;
                        MICRO_EDO_SIG<= MB;
                WHEN MB =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D1;
                        MICRO_EDO_SIG<= MC;
                WHEN MC =>
                        display_7seg <= LETRA_A;
                        anodos_7seg<=D2;
                        MICRO_EDO_SIG<= MD;
                WHEN MD =>
                        display_7seg <= LETRA_L;
                        anodos_7seg<=D3;
                        MICRO_EDO_SIG<= MA;
        END CASE;
WHEN H =>
        CASE MICRO_EDO_ACT IS
                WHEN MA =>
                        display_7seg <= LETRA_VACIO;
                        anodos_7seg<=D0;
                        MICRO_EDO_SIG<= MB;
                WHEN MB =>
                        display_7seg <= LETRA_A;
                        anodos_7seg<=D1;
                        MICRO_EDO_SIG<= MC;
                WHEN MC =>
                        display_7seg <= LETRA_L;
                        anodos_7seg<=D2;
                        MICRO_EDO_SIG<= MD;
                WHEN MD =>
                        display_7seg <= LETRA_O;
                        anodos_7seg<=D3;
                        MICRO_EDO_SIG<= MA;
```

```
                        END CASE;
                END CASE;
        END  PROCESS;

        PROCESS (clk,clr,en)  BEGIN
                IF (en = '1') then
                        if (clr = '1') then
                                MACRO_EDO_ACT<=A;
                                MICRO_EDO_ACT<=MA;
                                cont <=0;
                                cont2 <=0;
                        elsif (rising_edge(clk)) then
                                IF cont2 = 100000 THEN
                                        cont2<=0;
                                        MICRO_EDO_ACT<=MICRO_EDO_SIG;
                                ELSE
                                        cont2 <= cont2+1;
                                END  IF;

                                IF (cont = 100000000) THEN
                                        Ticks <=0;
                                        MACRO_EDO_ACT<=MACRO_EDO_SIG;
                                        MICRO_EDO_ACT<=MA;
                                ELSE
                                        cont <= cont+1;
                                END  IF;


                        end if;
                end if;
        END  PROCESS;
end  Marquesina;
```

V. Conclusión y Bibliografía

Conclusiones

Al término de la realización de esta práctica pudimos reforzar los conocimientos adquiridos en el curso de Diseño de Sistemas Digitales, más concretamente la parte de Máquina de Estados.

En este caso la Marquesina fue buena actividad para reforzar estos conocimientos.

Bibliografía

1.- https://definicion.de/marquesina/

2.- http://delta.cs.cinvestav.mx/~gmorales/ta/node50.html

3.- https://bloganalisis1.files.wordpress.com/2011/01/apuntesctsd10_mealy_moore.pdf