PRÁCTICA 7: MARQUESINA

Edgar A. Ramos Mesas 2013090243

ESCOM DSD 2CV10

MARQUESINA

Para el desarrollo de esta práctica se requirió el análisis de una máquina de estados un poco mas compleja que las vistas anteriormente, pues existían macro estados que estaban compuestos a su vez de varios estados. Los macro estados funcionaban a modo de control de parpadeo de los displays, ya que como sabemos, el display multiplexado no es capaz de mostrar diferentes caracteres al mismo tiempo, razón por la cual se requieren cambios rápidos entre cada estado, y un tiempo mayor para control de cada macro estado existente.

Se requirió del uso de un contador de 10 bits debido a que, como se mencionó anteriormente, cada macro estado tiene una duración mayor, además de que cada macro estado muestra diferentes caracteres en diferentes instantes de tiempo.

Dado que los bits de menor relevancia (LSB)en un contador cambian de forma más rápida en comparación con los mas significativos (MSB) es que se requiere del uso de los últimos, pues mientras no exista cambio en estos bits, el display estará oscilando entre los estados simples. En pocas palabras, los MSB le brindan a cada macro estado el tiempo suficiente para oscilar entre cada estado existente.

CODIGO IMPLEMENTADO.

Contador 10 bits.

Se hizo uso de un contador de 10 bits, cuyo código fue obtenido mediante la modificación del código existente del contador de 7 bits de la **Práctica 5**, sin embargo, para fines didácticos, se agrega a continuación el código desarrollado. Este código será grabado en la primer GAL22V10 empleada.

```
library ieee;
use ieee.std_logic_1164.all;

entity contador is port(
      clk, clr, en: in std_logic;
      q: inout std_logic_vector(9 downto 0)
);
end entity;

architecture a_contador of contador is
begin

process(clk, clr)
variable acc : std_logic;
begin
            if(clr = '1') then ---q(9) is MSB
                  q <= "0000000000";
            elsif(rising_edge(clk)) then
                  if (en = '0') then
                        q <= q;
                  else
                        for i in 0 to 9 loop
                              acc := '1';
                              for j in 0 to i-1 loop
                                    acc := acc and q(j);
                              end loop;
```

```
                        q(i) <= q(i) xor (en and acc);
                    end loop;
                end if;
            end if;
 end process;
 end architecture;
```

Marquesina

Para el desarrollo de la marquesina se deben tomar en cuenta los bits habilitadores de cada macro estado, dichos bits serán tomados de la salida del contador (bits 9 – 7) y serán agregados como entradas a la **segunda GAL22V10**, la cual contiene el código de control de la marquesina.

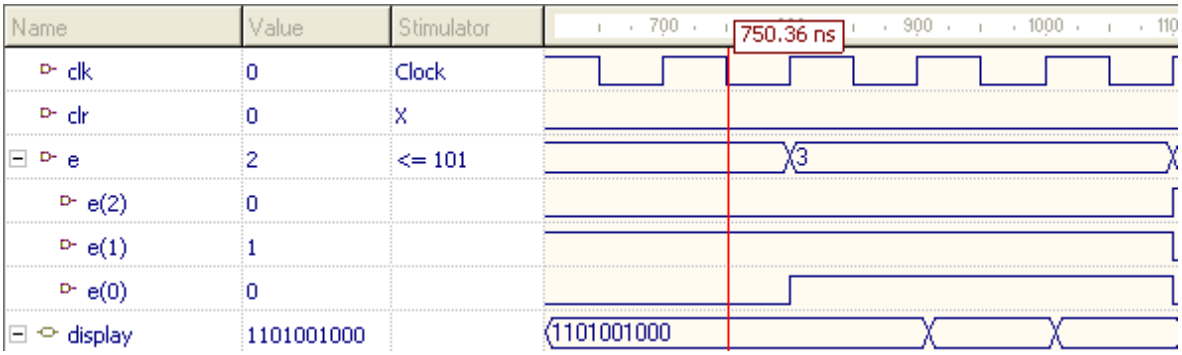
```
library ieee;
use ieee.std_logic_1164.all;

entity marquesina is port(
      clk, clr : in std_logic;
      e : in std_logic_vector(2 downto 0);
      display : inout std_logic_vector(9 downto 0)
);
end marquesina;

architecture a_marq of marquesina is

constant nd : std_logic_vector(2 downto 0) := "111";
constant d0 : std_logic_vector(2 downto 0) := "110";
constant d1 : std_logic_vector(2 downto 0) := "101";
constant d2 : std_logic_vector(2 downto 0) := "011";

constant nl : std_logic_vector(6 downto 0) := "1111111";
constant lH : std_logic_vector(6 downto 0) := "1001000";
constant lO : std_logic_vector(6 downto 0) := "0000001";
constant lL : std_logic_vector(6 downto 0) := "1110001";
constant lA : std_logic_vector(6 downto 0) := "0001000";

constant q0 : std_logic_vector(9 downto 0) := nd & nl;
constant q1 : std_logic_vector(9 downto 0) := d0 & lH;
constant q2 : std_logic_vector(9 downto 0) := d1 & lH;
constant q3 : std_logic_vector(9 downto 0) := d0 & lO;
constant q4 : std_logic_vector(9 downto 0) := d2 & lH;
constant q5 : std_logic_vector(9 downto 0) := d1 & lO;
constant q6 : std_logic_vector(9 downto 0) := d0 & lL;
constant q7 : std_logic_vector(9 downto 0) := d2 & lO;
constant q8 : std_logic_vector(9 downto 0) := d1 & lL;
constant q9 : std_logic_vector(9 downto 0) := d0 & lA;
constant q10 : std_logic_vector(9 downto 0) := d2 & lL;
constant q11 : std_logic_vector(9 downto 0) := d1 & lA;
constant q12 : std_logic_vector(9 downto 0) := d2 & lA;

begin
process(clk, clr)
begin
      if(clr = '1') then
            display <= q0;
```

```
elsif (rising_edge(clk)) then
      case display is
            when q0 =>
                  if (e = "000" or e = "001") then
                        display <= q0;
                  elsif (e = "010") then
                        display <= q1;
                  else
                        display <= "----------";
                  end if;
            when q1 =>
                  if (e = "010") then
                        display <= q1;
                  elsif (e = "011") then
                        display <= q2;
                  else
                        display <= "----------";
                  end if;
            when q2 =>
                  if (e = "011") then
                        display <= q3;
                  elsif (e = "100") then
                        display <= q4;
                  else
                        display <= "----------";
                  end if;
            when q3 =>
                  if(e = "011") then
                        display <= q2;
                  elsif (e = "100") then
                        display <= q4;
                  else
                        display <= "----------";
                  end if;
            when q4 =>
                  if(e = "100") then
                        display <= q5;
                  elsif (e = "101") then
                        display <= q7;
                  else
                        display <= "----------";
                  end if;
            when q5 =>
                  if(e = "100") then
                        display <= q6;
                  elsif (e = "101") then
                        display <= q7;
                  else
                        display <= "----------";
                  end if;
            when q6 =>
                  if(e = "100") then
                        display <= q4;
                  elsif (e = "101") then
                        display <= q7;
                  else
                        display <= "----------";
```

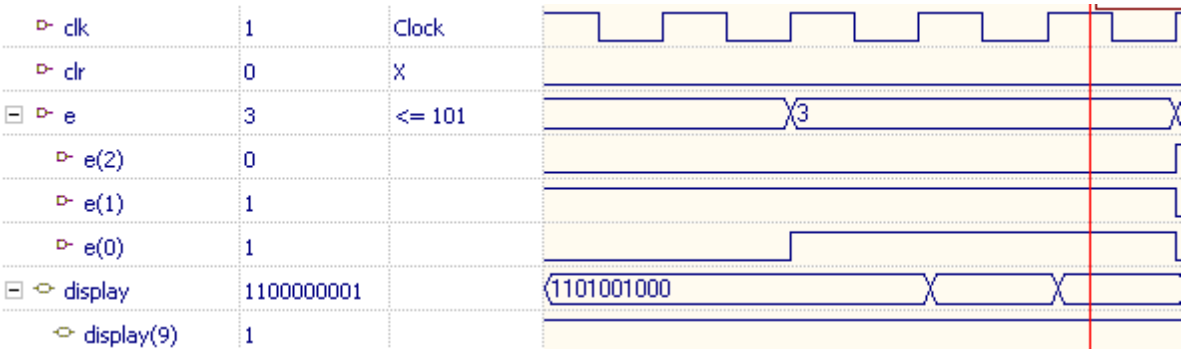
```
                        end if;
                  when q7 =>
                        if(e = "101") then
                              display <= q8;
                        elsif (e = "110") then
                              display <= q10;
                        else
                              display <= "----------";
                        end if;
                  when q8 =>
                        if(e = "101") then
                              display <= q9;
                        elsif (e = "110") then
                              display <= q10;
                        else
                              display <= "----------";
                        end if;
                  when q9 =>
                        if(e = "101") then
                              display <= q7;
                        elsif (e = "110") then
                              display <= q10;
                        else
                              display <= "----------";
                        end if;
                  when q10 =>
                        if(e = "110") then
                              display <= q11;
                        elsif (e = "111") then
                              display <= q12;
                        else
                              display <= "----------";
                        end if;
                  when q11 =>
                        if(e = "110") then
                              display <= q10;
                        elsif (e = "111") then
                              display <= q12;
                        else
                              display <= "----------";
                        end if;
                  when q12 =>
                        if(e = "111") then
                              display <= q12;
                        elsif (e = "000") then
                              display <= q0;
                        else
                              display <= "----------";
                        end if;
                  when others =>
                              display <= "----------";
                  end case;
      end if;
end process;
end a_marq;
```

SIMULACIONES.

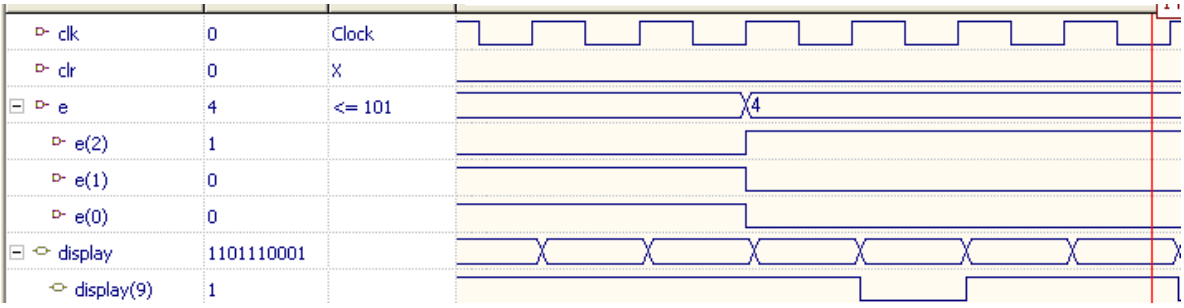
Para las simulaciones se manipularán los bits de entrada que controlan cada macro estado.



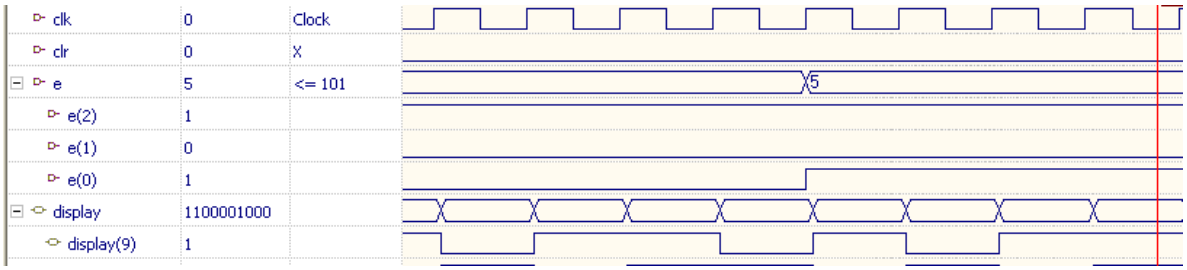
Se observa que en el primer estado los bits de control de macro estado “e” se encuentran establecidos en el primer valor que dicta una transición de estado, es decir, 010. Observamos además que los primeros 3 bits (display(9)- display(7)) son los bits de control de ánodo del display multiplexado. En este primer estado el display 0 muestra el código correspondiente a la primer letra del mensaje (HOLA) es decir que el código 1001000 corresponde a la letra H.



Se observa que con el cambio de macro estado existe también un cambio en la letra mostrada en el display 0, ´pues ahora el código asignado es 0000001 el cual corresponde a la segunda letra del mensaje, es decir la letra O.



En el siguiente cambio de macro estado observamos que de nueva cuenta el carácter ha cambiado, siendo esta vez el código mostrado el correspondiente a la tercera letra del mensaje, la letra L.



En este macro estado observamos un cambio que muestra el código correspondiente a la letra A, el código de la ultima letra del mensaje, lo cual permite darnos cuenta que, de hecho, el recorrido de la cadena a través de los displays esta siendo realizado de la manera que se esperaba, por lo cual se concluye que la práctica funciona de forma correcta.