Universidad Nacional Autonoma de Mexico Facultad de ingeniería

Diseño Digital VLSI

Grupo : 4

Práctica: 3

DISEÑO DEL CONTROL DE UN TREN ELÉCTRICO

Edgar Daniel Barcenas Martinez

OBJETIVO:
Demostrar a los estudiantes, mediante el diseño del sistema de control de un tren eléctrico, la forma de declarar tipos de datos diferentes a los definidos en el lenguaje VHDL.

ESPECIFICACIONES:
Diseñar un sistema digital que moverá un tren de derecha a izquierda y viceversa, sobre una línea, deteniéndose en cada estación por 2 minutos. En cada estación hay unos sensores que detectan cuando un tren entra a una estación. Existe un botón de emergencia en los vagones que hará que el tren se detenga por un minuto de más en la estación, si así se requiere.

DIAGRAMA DE BLOQUES:

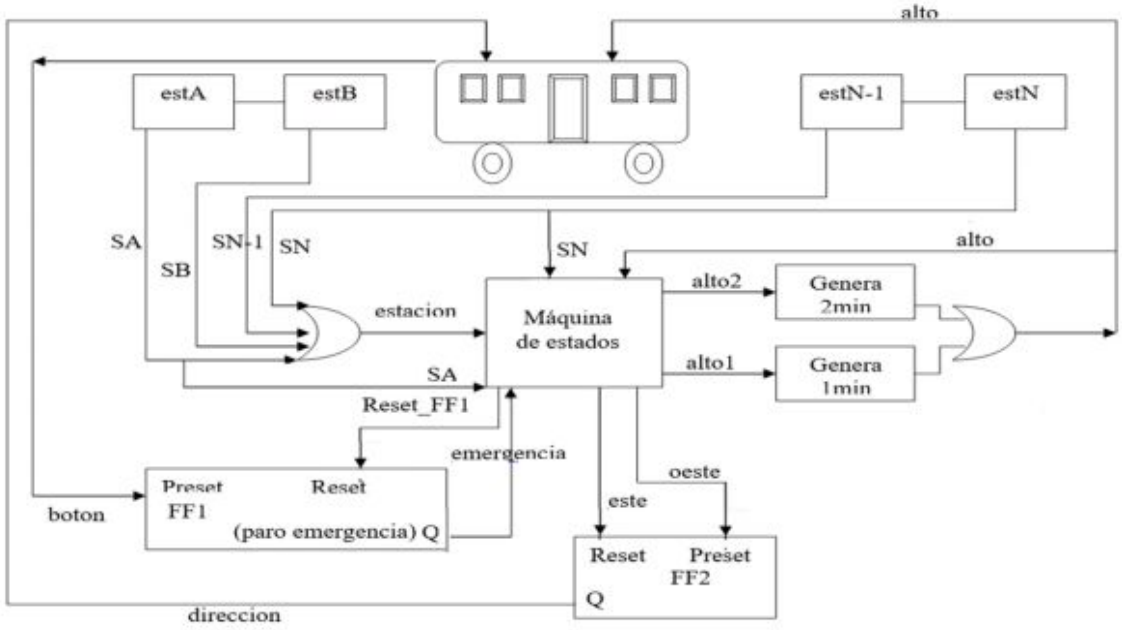

Figura 3.1. Diagrama de bloques del sistema Tren Eléctrico

La figura 3.2 muestra los bloques funcionales dentro del FPGA del sistema Tren Eléctrico y en la figura 3.3 se muestra su Carta ASM.

BLOQUES FUNCIONALES:

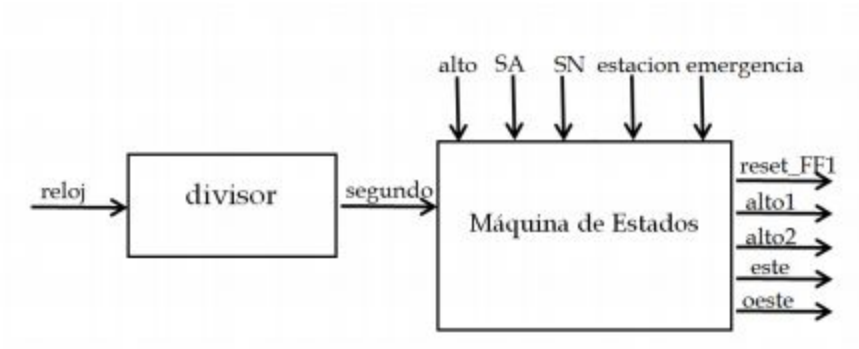

Figura 3.2. Bloques funcionales del sistema tren Eléctrico

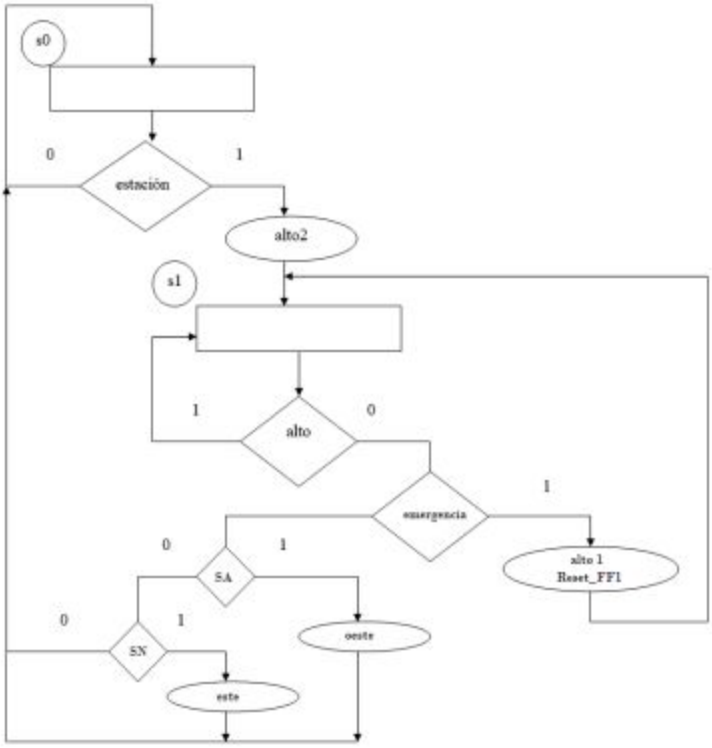

Figura 3.3. Carta ASM del sistema Tren Eléctrico

Actividad de la Clase:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity tren is
   Port (reloj, alto, SA, SN, emergencia, estacion : in std_logic;
   alto1, alto2, este, oeste, reset_FF1: out std_logic);
end tren;
```

Figura 3.4. Entidad del sistema Tren Eléctrico

```
architecture Behavioral of tren is
   signal segundo : std_logic;
   type estados is (s0, s1);
   signal epresente, esiguiente : estados;
```

La figura 3.5 muestra la parte declaratoria de la arquitectura en el sistema Tren Eléctrico.

```
begin
   process (reloj)
   begin
      if rising_edge(segundo) then
         epresente <= esiguiente;
      end if;
   end process;

   process ( epresente, estacion, alto, emergencia, SA,SN)
   begin
      case epresente is
         when s0 =>
            if estacion  = '1' then
               alto2 <= '1';
               esiguiente <=s1;
```

```
            elsif
               esiguiente <=s0;
            end if;
         when s1 =>
            if alto  = '1' then
               esiguiente <=s1;
            elsif emergencia = '1' then
               reset_FF1 <= '1';
               alto1 <= '1';
               esiguiente <=s1;
            elsif SA = '1' then
               oeste <= '1';
               esiguiente <=s0;
            elsif SN = '1' then
               este <= '1';
               esiguiente <=s0;
            elsif  esiguiente <=s0;
            end if;
   end process;
end Behavioral;
```

Figura 3.6. (continuación) Parte operatoria de la arquitectura del sistema Tren Eléctrico

Programa Final:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity p3 is
   Port ( SA : in  STD_LOGIC;
          SN : in  STD_LOGIC;
          S2 : in  STD_LOGIC;
          S3 : in  STD_LOGIC;
          emergencia : in  STD_LOGIC;
          este : out  STD_LOGIC;
          oeste : out  STD_LOGIC;
          avanza : out  STD_LOGIC;
          detiene : out  STD_LOGIC);
end p3;

architecture Behavioral of p3 is
          signal segundo : std_logic;
          signal alto1,alto2,alto,reset_FF1,estacion: std_logic
          type estados is (s0, s1);
```

```
        signal epresente, esiguiente : estados;
begin
estacion := SA = '1' OR S1 = '1' OR S2 = '1' OR SN = '1';

divisor : process (reloj)
variable CUENTA: STD_LOGIC_VECTOR(27 downto 0) := X"0000000";
  begin
                if rising_edge(reloj) then
                        if CUENTA = X"48009E0" then
                                CUENTA := x"0000000";
                        else
                                CUENTA := CUENTA+1;
                        end if;
                end if;
                        segundo <= CUENTA(22);
        end process;

process (reloj) begin
        if rising_edge(segundo) then
                epresente <= esiguiente;
        end if;
end process;

process ( epresente, estacion, alto, emergencia, SA,SN)
        begin
                case epresente is
                        when s0 =>
                        if estacion = '1' then
                                alto2 <= '1';
                                esiguiente <=s1;
                        else
                                esiguiente <=s0;
                        end if;
                        when s1 =>
                                alto2 <= '0';
                                if alto = '1' then
                                        esiguiente <=s1;
                                        alto1 <= '0';
                                        reset_FF1 <= '0';
```

```
                elsif emergencia = '1' then
                        reset_FF1 <= '1';
                        alto1 <= '1';
                        esiguiente <=s1;
                elsif SA = '1' then
                        oeste <= '1';
                        esiguiente <=s0;
                        este <= '0';
                elsif SN = '1' then
                        este <= '1';
                        oeste <= '0';
                        esiguiente <=s0;
                else
                        esiguiente <=s0;
                end if;
        end case;
end process;
end Behavioral;
```

Conclusiones:

Se demostró mediante el diseño del sistema de control de un tren eléctrico, la forma de declarar tipos de datos diferentes a los definidos en el lenguaje VHDL.

Diseñar un sistema digital que moverá un tren de derecha a izquierda y viceversa, sobre una línea, deteniéndose en cada estación por 2 minutos. En cada estación hay unos sensores que detectan cuando un tren entra a una estación. Existe un botón de emergencia en los vagones que hará que el tren se detenga por un minuto de más en la estación, si así se requiere.