Prácticas de diseño

Utilizando dispositivos lógicos programables

Norma Elva
Chávez Rodríguez
Jorge
Valeriano Assem

PRÁCTICAS DE DISEÑO

UTILIZANDO DISPOSITIVOS LÓGICOS PROGRAMABLES

FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE INGENIERÍA EN COMPUTACIÓN.

ING. NORMA ELVA CHAVEZ RODRIGUEZ
ING. JORGE VALERIANO ASSEM

CIUDAD UNIVERITARIA, JULIO 2002

PRESENTACIÓN

La Facultad de Ingeniería ha decidido realizar una serie de ediciones provisionales de obras recientemente elaboradas por académicos de la institución, como material de apoyo para sus clases, de manera que puedan ser aprovechadas de inmediato por alumnos y profesores. Tal es el caso de la obra *Prácticas de diseño utilizando dispositivos lógicos programables*, de los profesores Norma Elva Chávez Rodríguez y Jorge Valeriano Assem.

Se invita a los estudiantes y profesores a que comuniquen a los autores las observaciones y sugerencias que mejoren el contenido de la obra, con el fin de que se incorporen en una futura edición definitiva.

INDICE:

INTRODUCCIÓN:

Las presentes prácticas son una herramienta de gran utilidad para todos aquellos que quieran diseñar proyectos con dispositivos lógicos programables, especialmente utilizando el paquete de MAX+PLUS II.

El objetivo principal de este trabajo es aportar a la comunidad de la facultad de Ingeniería UNAM , un material de apoyo para motivar tanto a alumnos como profesores el uso de dispositivos lógicos programables.

Es importante señalar que para entender estas prácticas es necesario tener claro los conceptos de lenguajes HDL y el paquete de MAX+PLUS II.

INDICE

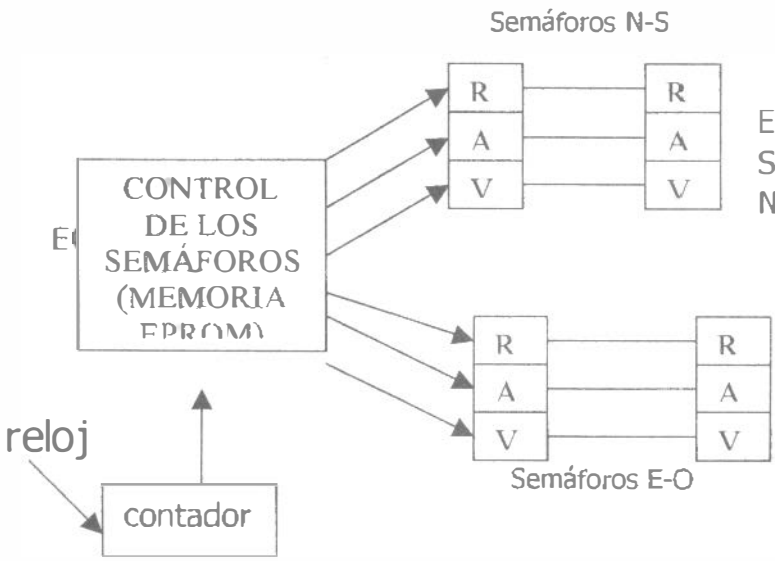
CONTROL DE SEMÁFOROS DEL CRUCE DE DOS AVENIDAS

OBJETIVO:
Diseñar un sistema que controle el cruce de dos avenidas, utilizando una memoria EPROM.
ESPECIFICACIONES:

- Habrá cuatro semáforos en cuatro distintas direcciones.
- Los semáforos (N-S) y (E-O) se pondrán en rojo al mismo tiempo
- (N-S) y (E-O) nunca tendrán la luz verde o amarilla al mismo tiempo.
- Las luces verdes durarán 45 segundos.
- Las luces amarillas durarán 15 segundos.
- Las luces rojas durarán 60 segundos.

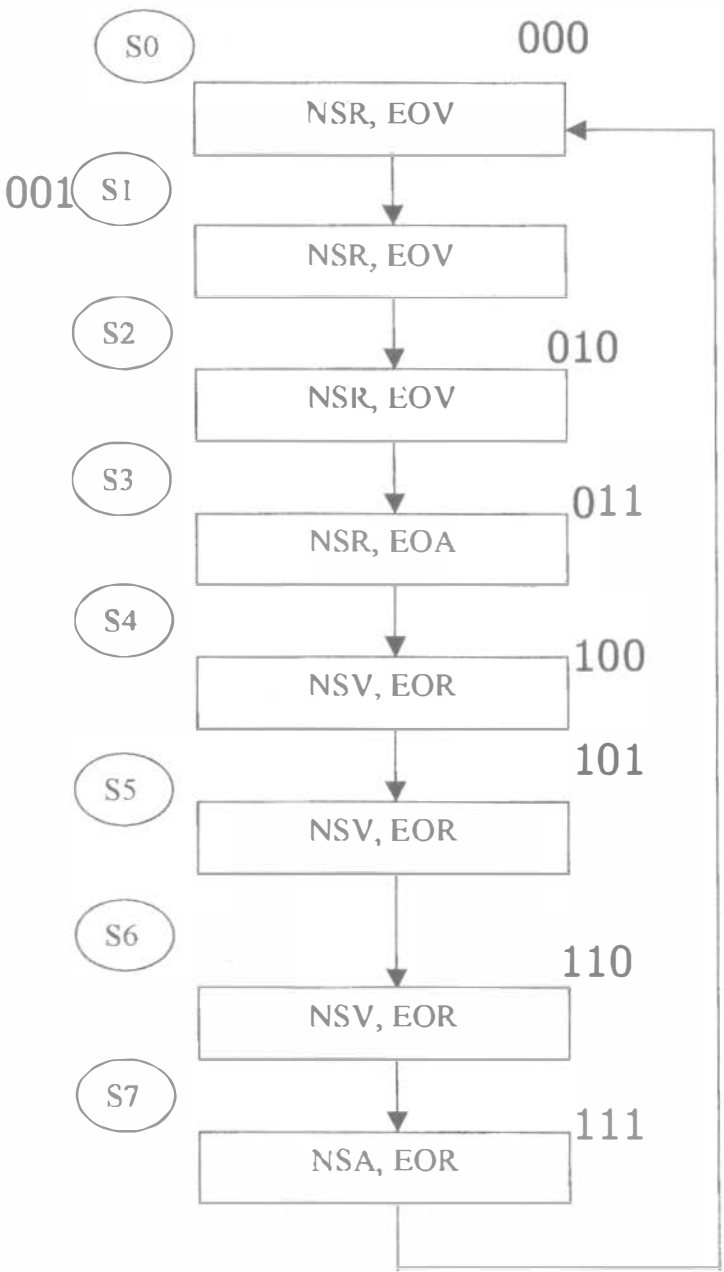
DESARROLLO:

Lo primero será hacer un diagrama de bloques para visualizar mejor el problema:



ENTRADAS: RELOJ.
SALIDAS:
NSR, NSA, NSV, EOR, EOA,

Se requiere que sea un solo reloj para todo el diseño,
Pero debido a que las salidas manejan distintos tiempo se hace un divisor de tiempo, en este caso será de 15 ya que es el tiempo menor y los otros tiempos son divisibles entre 15.
Posteriormente se hace la carta ASM y la asignación binaria a los estados de nuestro control:

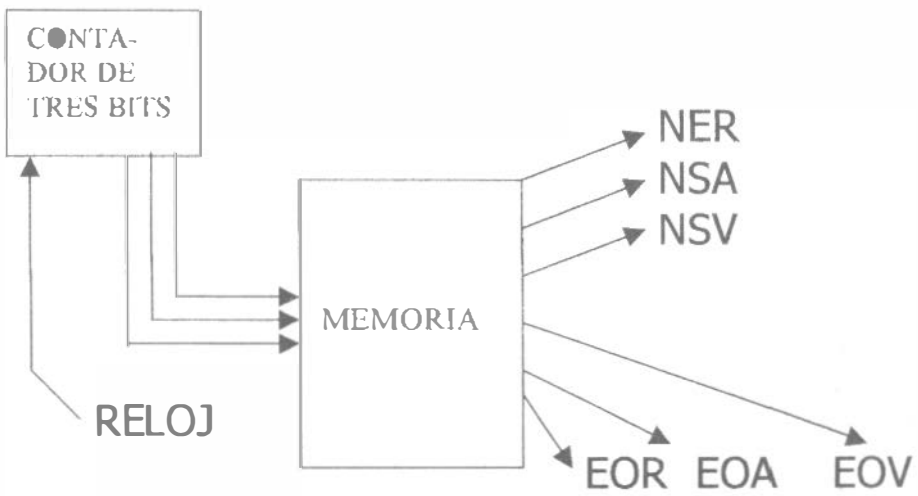


POR LO TANTO SE REQUIERE DE UNA MEMORIA QUE TENGA 8 REGISTROS y CADA REGISTRO DE 8 BITS:

Y EL CONTENIDO DE LA MEMORIA SERÀ:

DIRECCIONES ESTADO PRESENTE	CONTENIDO NSR	NSA	NSV	EOR	EOA	EOV		
000	1	0	0	0	0	1	0	0
001	1	0	0	0	0	1	0	0
010	1	0	0	0	0	1	0	0
011	1	0	0	0	1	0	0	0
100	0	0	1	1	0	0	0	0
101	0	0	1	1	0	0	0	0
110	0	0	1	1	0	0	0	0
111	0	1	0	1	0	0	0	0

IMPLEMENTANDO:



INDICE

SISTEMA DE ALARMA PARA UNA CASA

OBJETIVO:

Diseñar un circuito básico que ilustre el uso del editor gráfico de MAX+plus II.

Diseñar un sistema de alarma para detectar intrusos, en puntos estratégicos alrededor de la casa habitación.

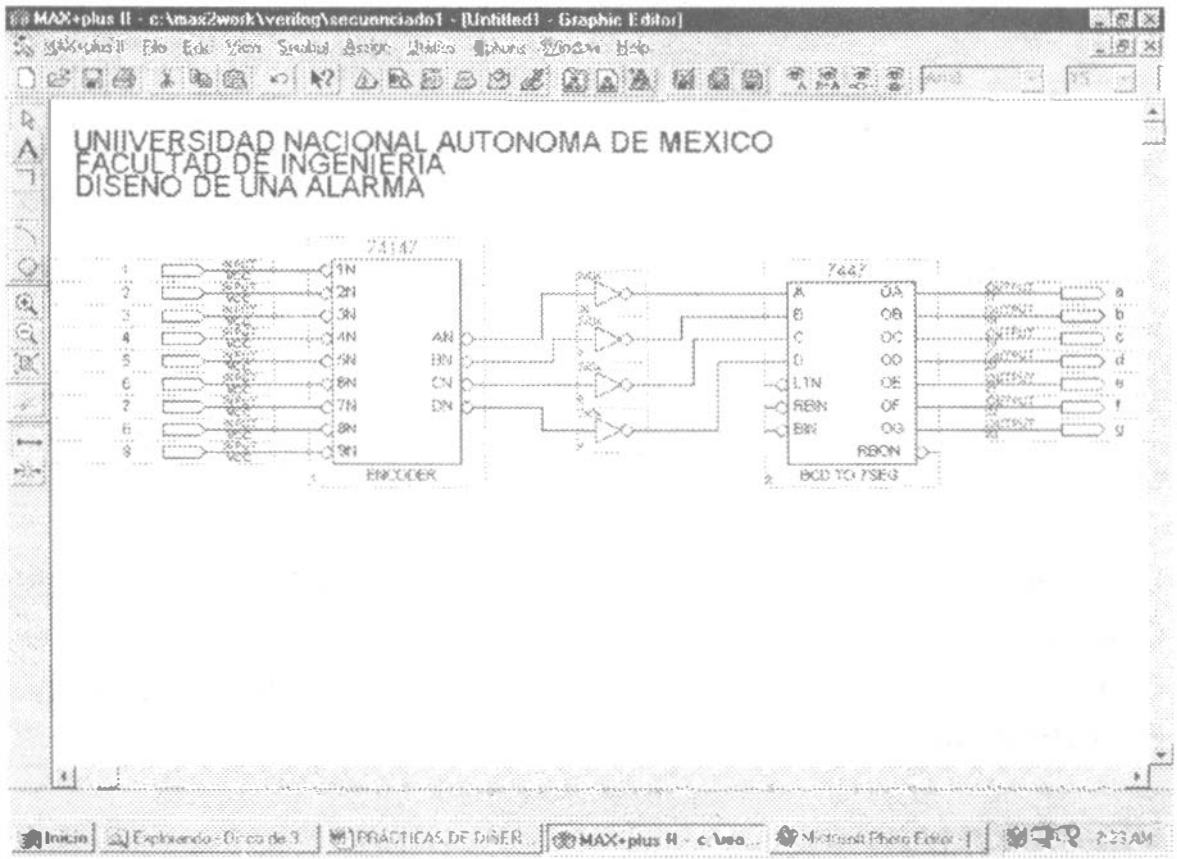
ESPECIFICACIONES:

La alarma contiene nueve interrupciones o sensores distribuidos en distintos puntos de una casa. Al accionarse alguno de los sensores el circuito mostrará en un display el sensor que activo la alarma.

DESARROLLO:

Podrá hacerse uso de fotoresistencias o sensores ópticos con los cuales se accionará la alarma, por lo que éstas serán las entradas a nuestro control. Estas entradas deberán ser numeradas del uno al nueve con el fin de que puedan ser reconocidas al accionarse la alarma y se mostrarán en el display.

Todas estas entradas-sensores estarán conectadas a un decodificador 74LS147. Las salidas que se tienen del decodificador se conectan a un inversor (74LS04) cada una de ellas y a su vez éstas entrarán a un convertidor a BCD a siete segmentos.



SISTEMA DE SEGURIDAD PARA UNA CASA USANDO EL EDITOR GRÁFICO.

INDICE

SISTEMA DE ALARMA PARA UNA CASA USANDO LENGUAJE DE PROGRAMACIÓN AHDL

OBJETIVO

Diseñar un circuito básico que ilustre el uso del editor de texto y el lenguaje AHDL

Diseñar un sistema de alarma para detectar intrusos, en puntos estratégicos alrededor de la casa habitación.

ESPECIFICACIONES:

La alarma contiene nueve interrupciones o sensores distribuidos en distintos puntos de una casa. Al accionarse alguno de los sensores el circuito mostrara en un display el sensor que activo la alarma.

Podrá hacerse uso de fotoresistencias o sensores ópticos con los cuales accionar la alarma, por lo que estas serán las entradas a nuestro circuito. Estas entradas deberán ser numeradas del uno al nueve con el fin de que puedan ser reconocidas al accionarse la alarma y se mostraran en el display.

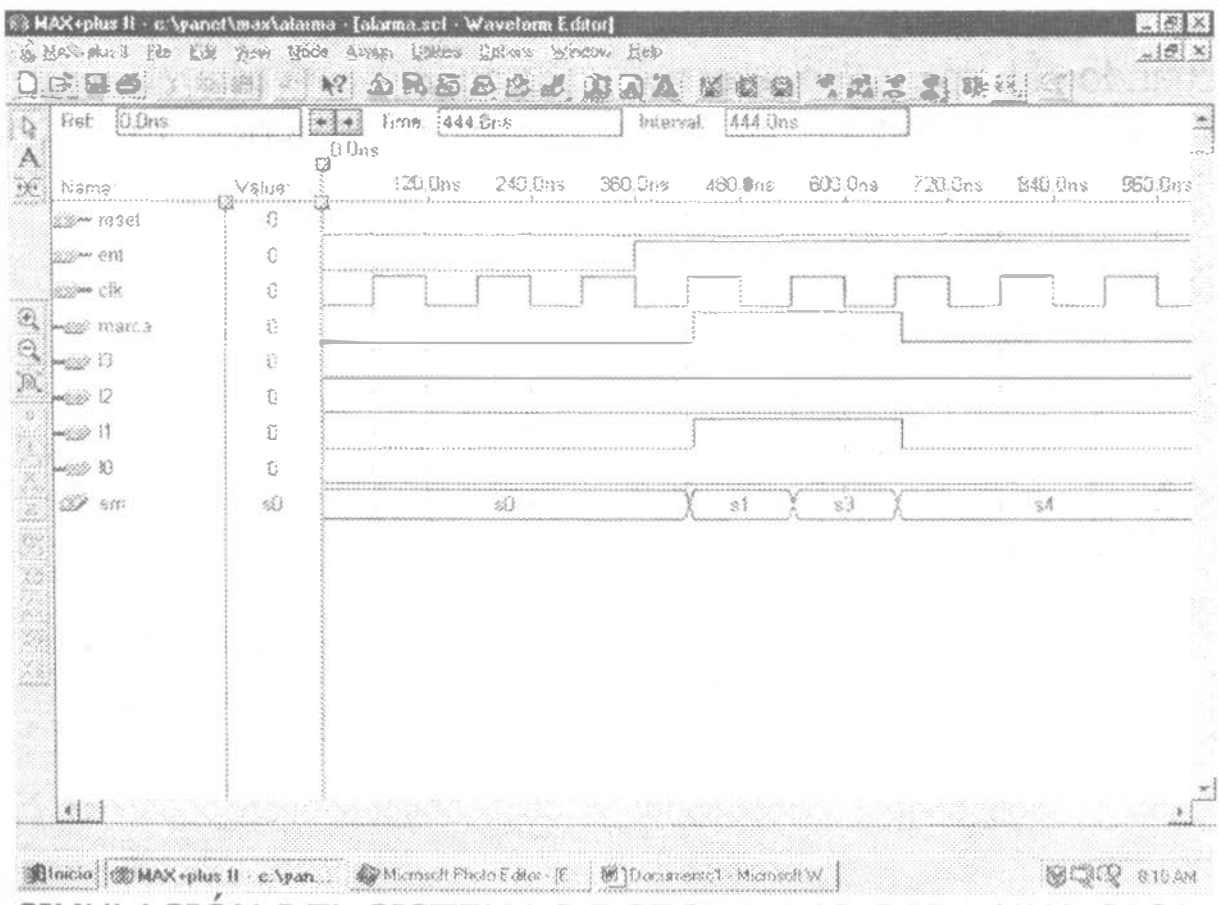
Todas estas entradas-sensores estarán conectadas a un decodificador (74LS147). Las salidas que se tienen del decodificador se conectan a un inversor (74LS04) cada una de

ellas y a su vez éstas entrarán a un convertidor a BCD a siete segmentos.

UTILIZANDO EL LENGUAJE AHDL:

```
TITLE"alarma";
SUBDESIGN alarma(clk, reset,ent: input;
I[3..0], marca:OUTPUT;)
VARIABLE
sm: MACHINE OF  BITS  (b2, b1,b0)
WITH STATES (
s0=b"000",
s1=b"001",
s2=b"010",
s3=b"011",
s4=b"100");
BEGIN
DEFAULTS
I[3..0]=GND;
END DEFAULTS;
sm.clk=clk;
sm.reset=reset;
CASE sm IS
WHEN  s0=>
IF ent THEN
sm=s1;
ELSE
sm=s0;
END IF;
WHEN s1=>
```

```
sm=s2;
I1=VCC;
marca=VCC;
sm=s3;
WHEN s3=>
marca= VCC;
I1=VCC;
sm=s4;
END CASE;
END;
```



SIMULACIÓN DEL SISTEMA DE SEGURIDAD PARA UNA CASA

SENSADO EN UN SISTEMA DE INSTRUMENTACIÒN

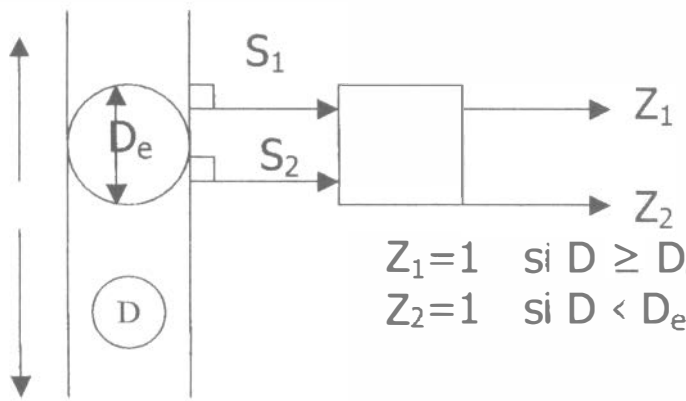
OBJETIVO:

Con esta práctica se comenzará a tener una mejor comprensión del concepto de Cartas ASM mediante el problema descrito en el Desarrollo de esta práctica.

ESPECIFICACIONES:

En un sistema de instrumentación, se requiere saber si un balón de un diámetro mayor a un cierto valor específico, se mueve hacia arriba o hacia abajo dentro de un tubo. El tubo tiene un diámetro mínimo requerido del balón. Cada sensor entrega a la salida un 1 lógico cuando se detecta la presencia del balón y un 0 lógico sino. Se requieren dos salidas: z1=1 cuando el balón del diámetro requerido va hacia arriba,
Y z2=1 cuando el balón del diámetro requerido va hacia abajo.

DIAGRAMA DE BLOQUES:



$Z_1=1$ si $D \geq D_e$ y el balón va hacia arriba
$Z_2=1$ si $D < D_e$ y el balón va hacia abajo.

CARTA ASM:

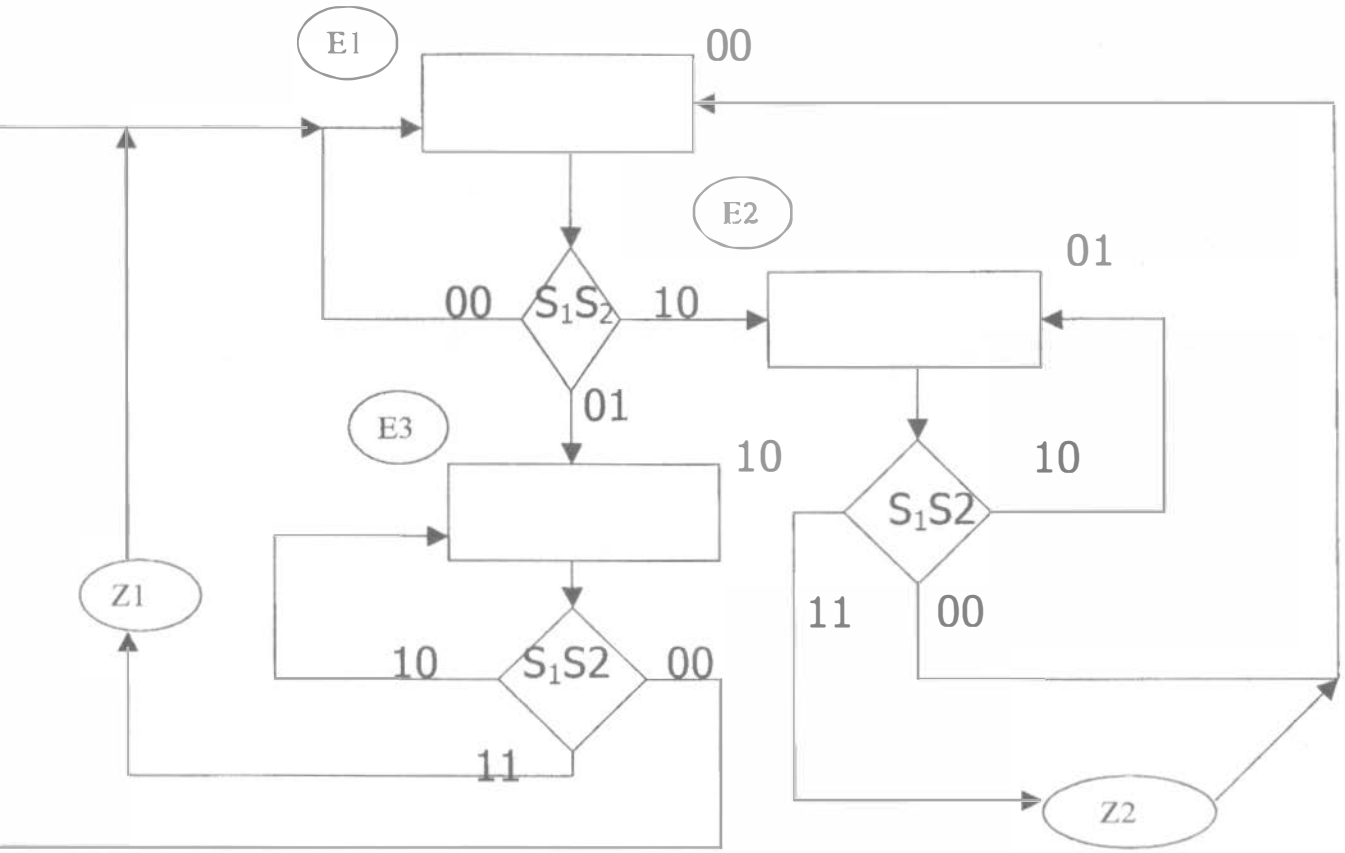


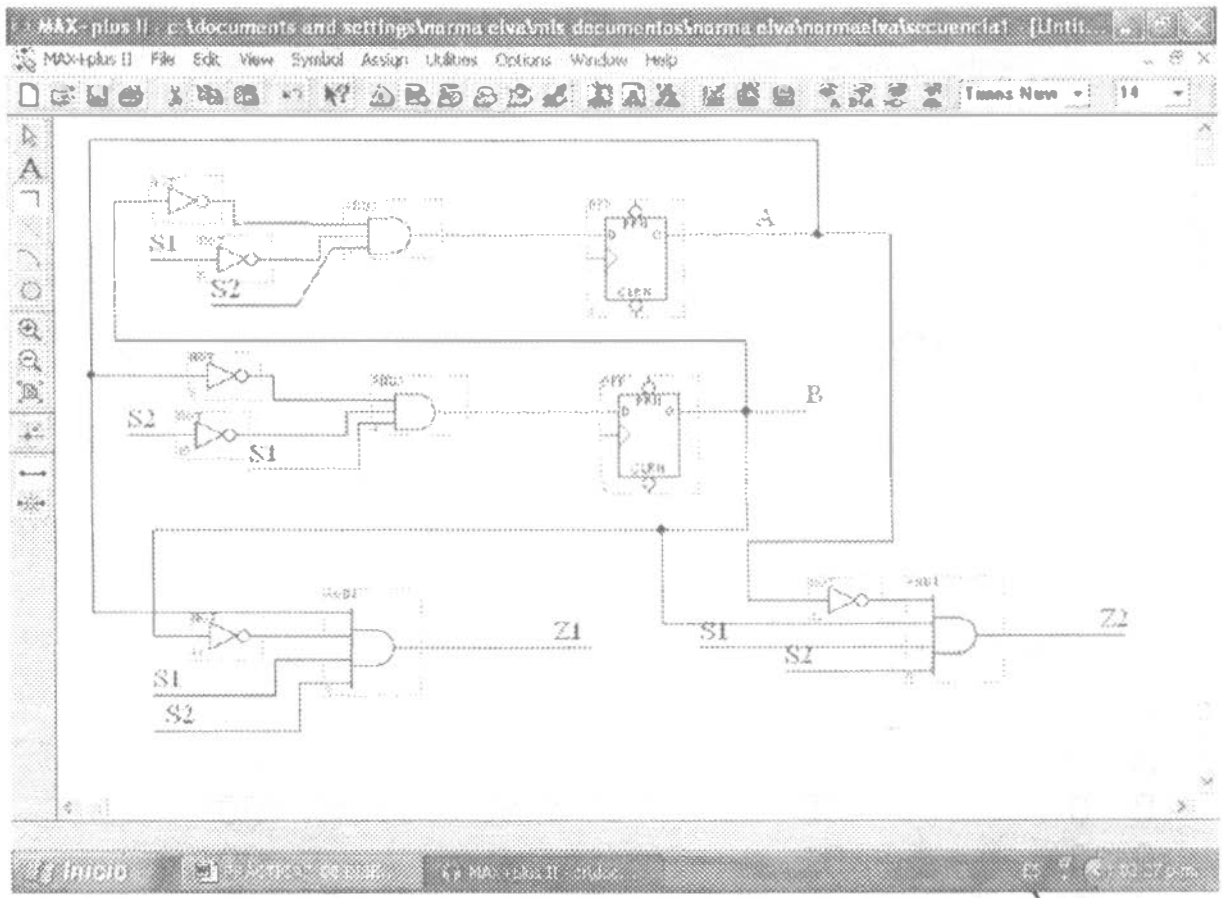
TABLA DE TRANSICIÓN DE ENTRADAS:

Estado presente		Entrada	Estado siguiente	salidas
A	B	S1S2	A^+B^+	Z_1Z_2
0	0	00	00	00
0	0	01	10	00
0	0	10	01	00
0	1	00	00	00
0	1	10	01	00
0	1	11	00	01
1	0	00	00	00
1	0	01	10	00
1	0	11	00	10

Se requieren dos flip-flops "D" y las ecuaciones son:

$D_A = B' S'_1 S_2$
$D_B = A' S_1 S'_2$

$Z_1 = AB'S_1S_2$
$Z_2 = A'BS_1S_2$



SISTEMA DE SENSADO UTILIZANDO EL EDITOR GRÁFICO

INDICE

MÁQUINA DESPACHADORA DE REFRESCO

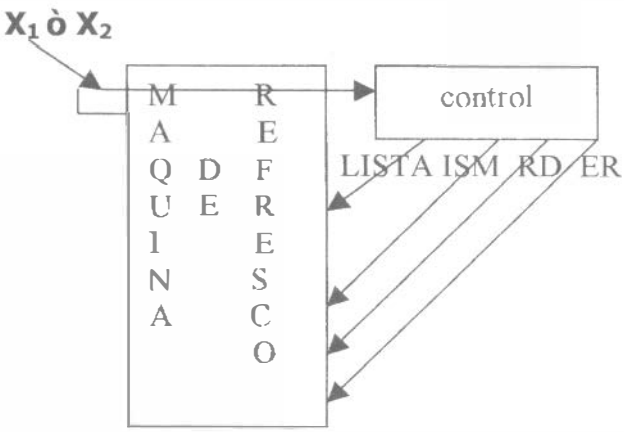
OBJETIVO:
Hacer el controlador de una máquina de refrescos utilizando el lenguaje AHDL.
ESPECIFICACIONES:
Costo del refresco $10.00. Acepta únicamente monedas de $5.00 y $10.00, solo acepta cantidades correctas, regresa cantidades incorrectas.
Nota: únicamente acepta de una en una las monedas.

Solución:

Entradas: X_1 = $10.00 , X_2 = $5.00
Existen dos tipos de salidas:
Activadores: ER = Entrega Refresco.
RD = Regresa Dinero.

Indicadores: Lista = Máquina lista para operar.
ISM = Inserta siguiente moneda.

DIAGRAMA DE BLOQUES:

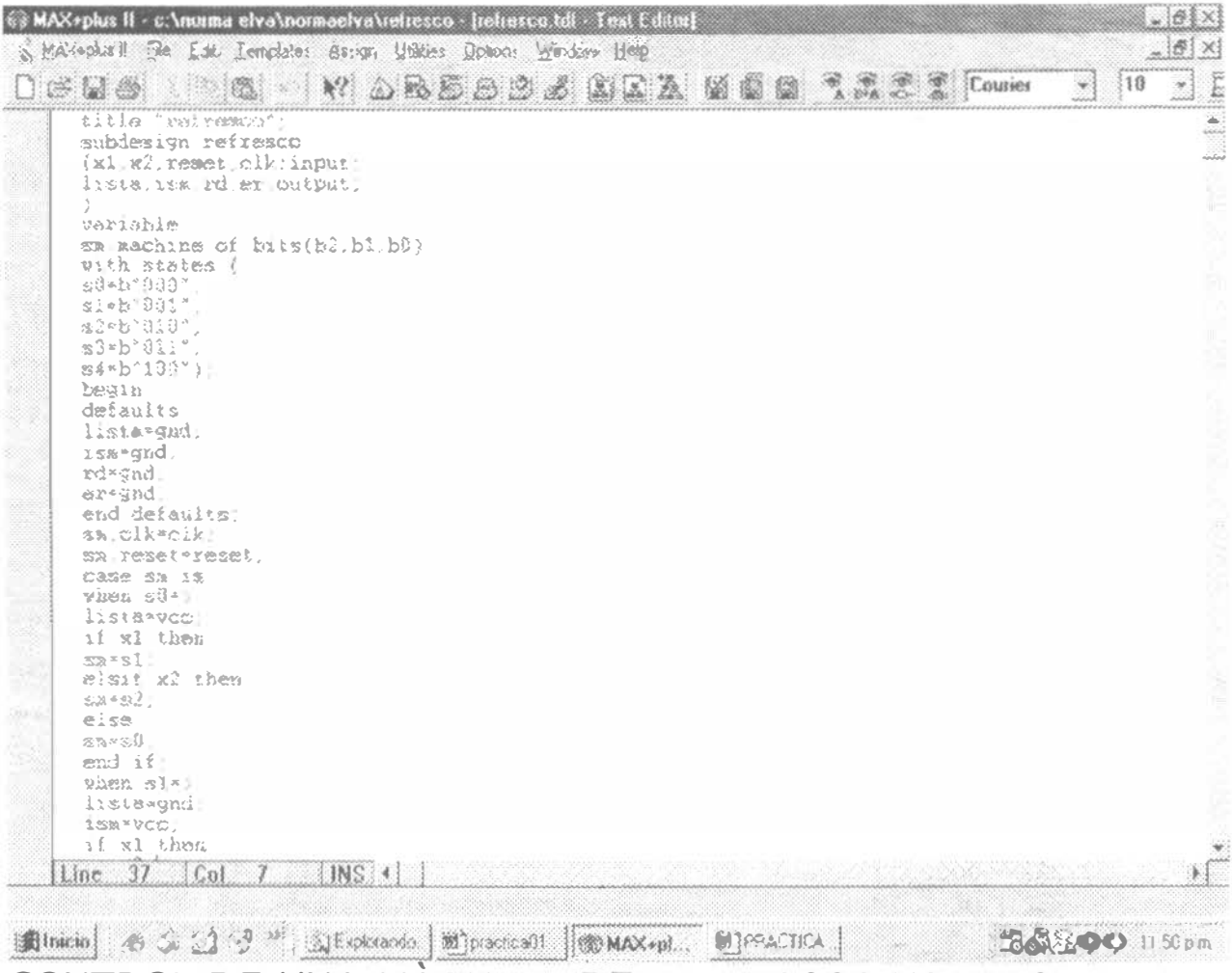


DESARROLLO:

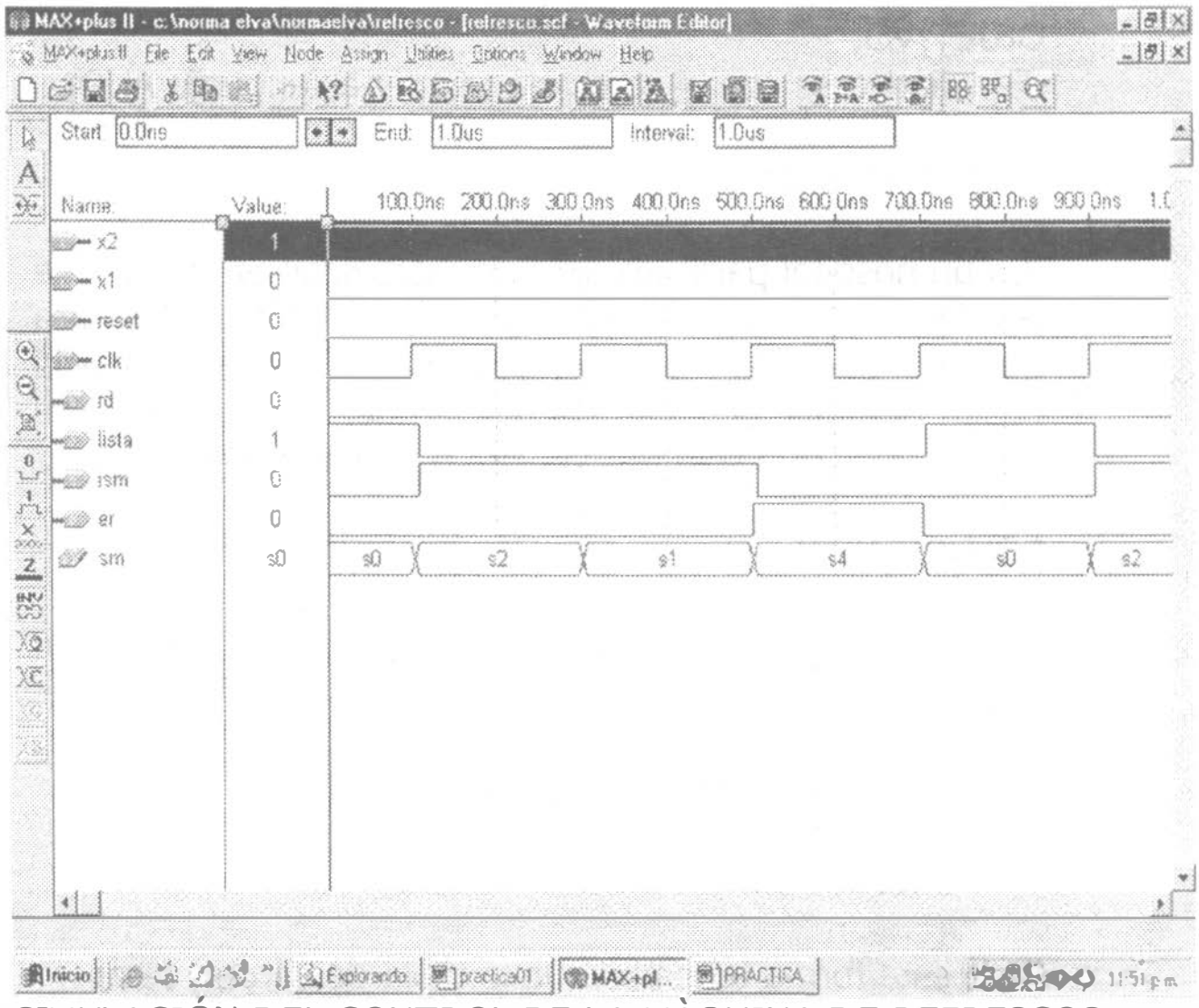
Utilizando lenguaje AHDL:

```
title "refresco";
subdesign refresco
(x1,x2,reset,clk:input;
lista,ism,rd,er:output;
)
variable
sm:machine of bits(b2,b1,b0)
with states (
s0=b"000",
s1=b"001",
s2=b"010",
s3=b"011",
s4=b"100");
begin
defaults
lista=gnd;
ism=gnd;
rd=gnd;
er=gnd;
end defaults;
sm.clk=clk;
sm.reset=reset;
case sm is
when s0=>
lista=vcc;
if x1 then
sm=s1;
elsif x2 then
```

```
            sm=s2;
            else
            sm=s0;
            end if;
when s1=>
            lista=gnd;
            ism=vcc;
            if x1 then
            sm=s3;
            elsif x2 then
            sm=s4;
            else
            sm=s1;
            end if;
            when s2=>
            ism=vcc;
            if x1 then
            sm=s4;
            elsif x2 then
            sm=s1;
            else
            sm=s2;
            end if;
            when s3=>
            ism=gnd;
            rd=vcc;
            sm=s0;
            when others=>
            rd=gnd;
            er=vcc;
            sm=s0;
            end case;
            end;
```



CONTROL DE UNA MÀQUINA DE REFRESCOS USANDO AHDL Y MAX+PLUS II



SIMULACIÓN DEL CONTROL DE LA MÀQUINA DE REFRESCOS

INDICE

SISTEMA DEL CONTROL DE ENTRADAS A UN HOSPITAL en AHDL.

OBJETIVO:

Familiarizarse con el editor de texto y el lenguaje AHDL

ESPECIFICACIONES:

En un hospital para que los pacientes sean atendidos, se les da una ficha y se requiere hacer el controlador para que los pacientes que tengan una super emergencia pasen primero, después los que tengan una emergencia y al final los que sólo tengan cita para chequeo.

SOLUCIÓN:

Se tiene tres posibles entradas, así que haremos un asignación binaria a esas entradas.

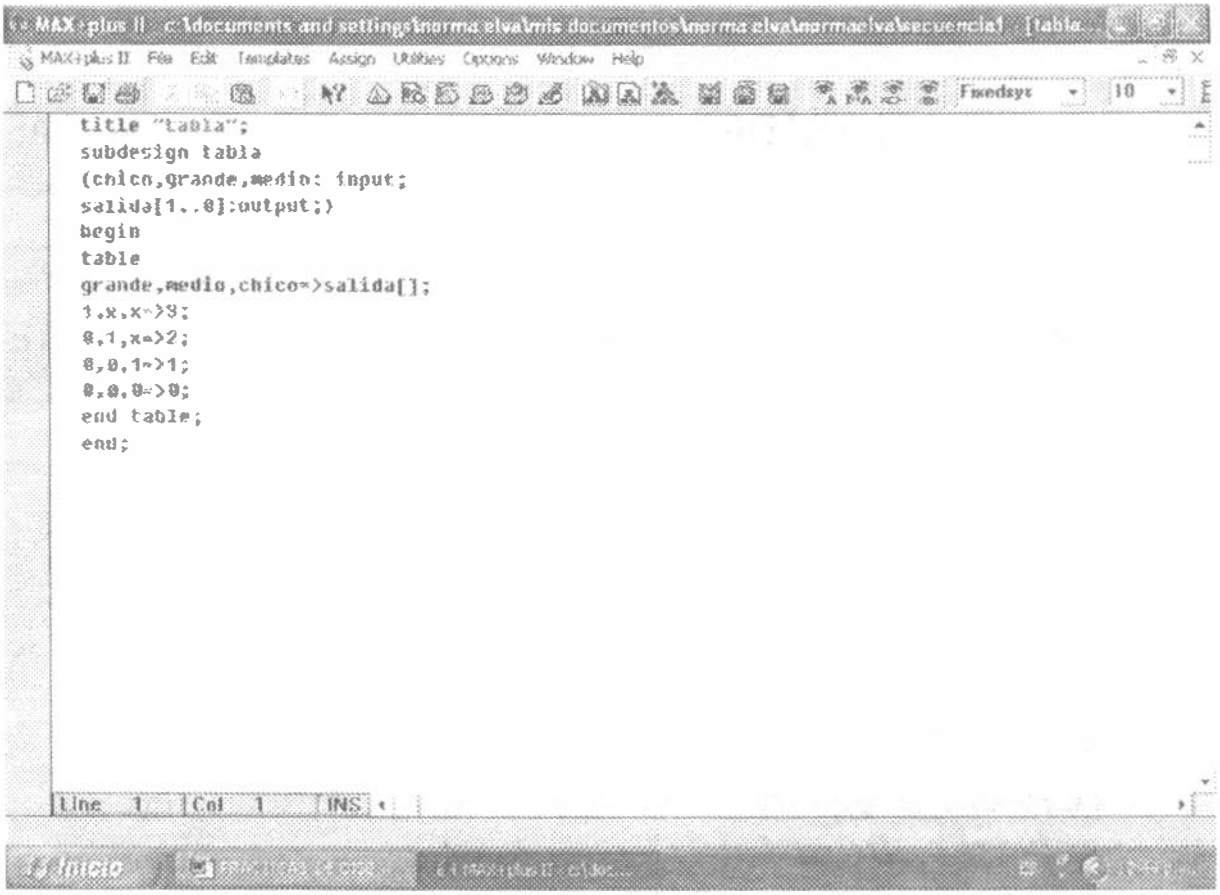
Super emergencia (SE)... = 1 1
Emergencia (E)..... = 1 0
Chequeo (CH)..... = 0 1

Y de esta forma tenemos que la tabla de verdad será:

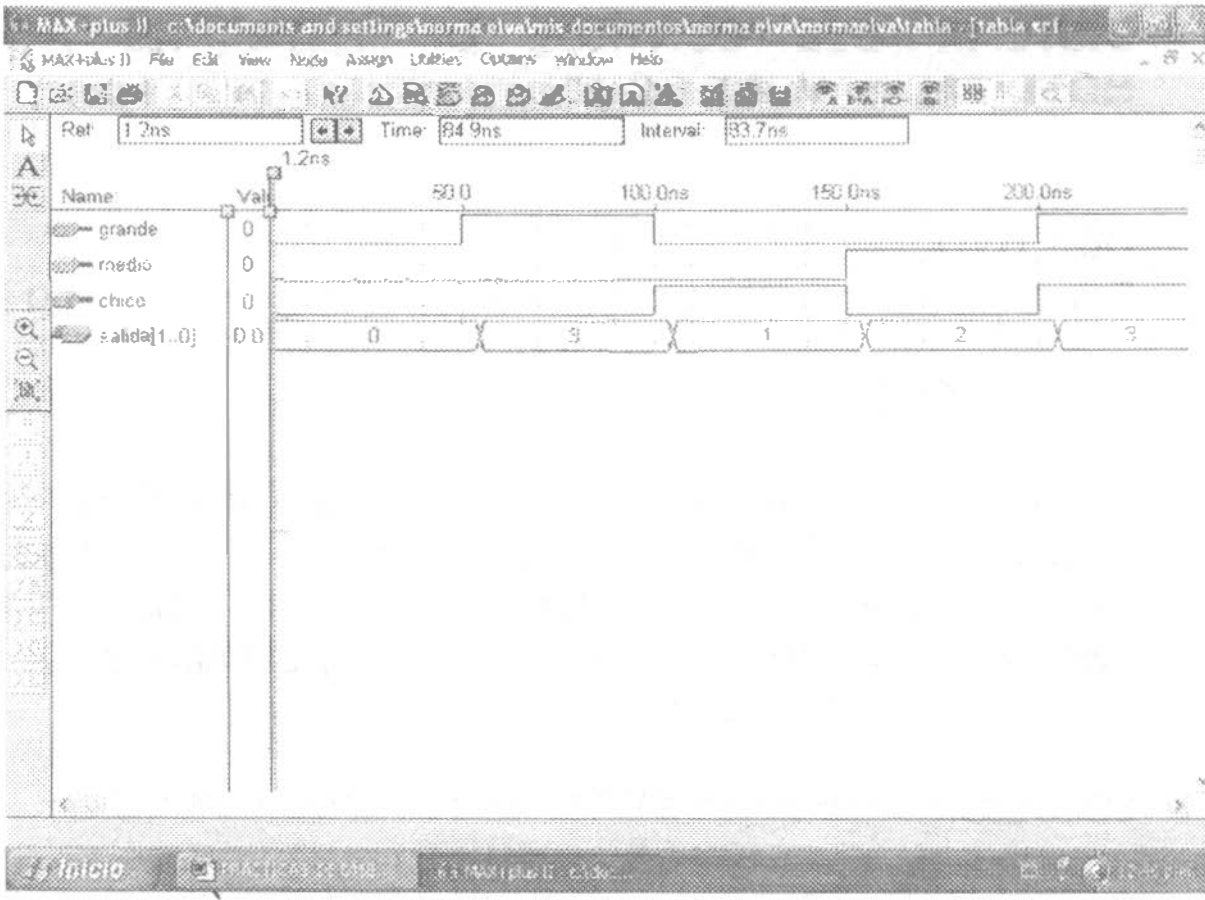
ENTRADAS			SALIDAS
SE	E	CH	Turno
1	*	*	1 1
0	1	*	1 0
0	0	1	0 1
0	0	0	0 0

Por lo tanto introduciendo la tabla al lenguaje AHDL:

```
title "tabla";
subdesign tabla
(CH,E,SE: input;
salida[1..0]:output;)
begin
table
SE,E,CH=>salida[];
1,x,x=>3;
0,1,x=>2;
0,0,1=>1;
0,0,0=>0;
end table;
end;
```



CONTROL DE ENTRADAS A UN HOSPITAL (AHDL)



SIMULACIÒN DEL CONTROL DE ENTRADAS A UN HOSPITAL

INDICE

SISTEMA DEL CONTROL DE ENTRADAS A UN HOSPITAL en VERILOGHDL.

OBJETIVO:
Familiarizarse con el editor de texto y el lenguaje Verilog HDL

ESPECIFICACIONES:

En un hospital para que los pacientes sean atendidos, se les da una ficha y se requiere hacer el controlador para que los pacientes que tengan una super emergencia pasen primero, después los que tengan una emergencia y al final los que solo tengan cita para chequeo.

SOLUCIÓN:

Se tiene tres posibles entradas, así que haremos un asignación binaria a esas entradas.

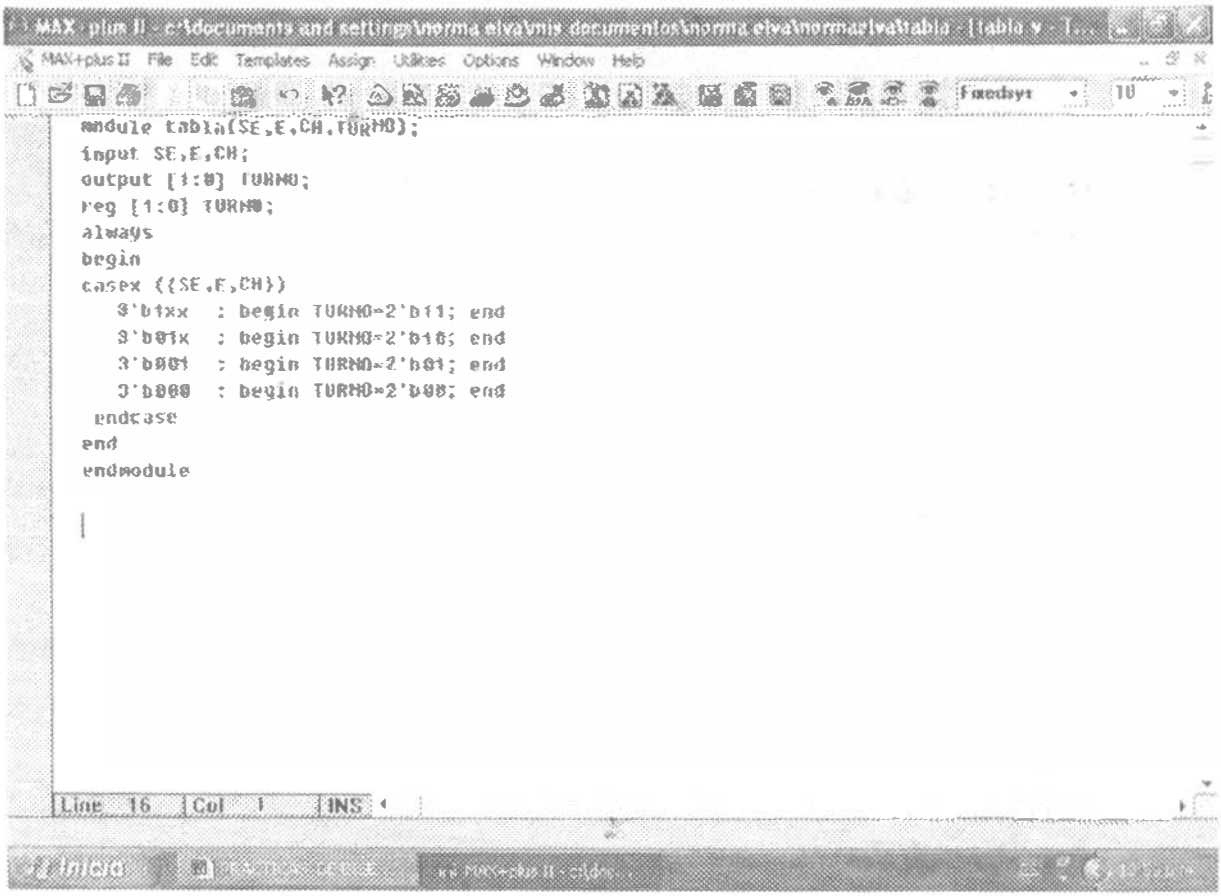
Super emergencia (SE)... = 1 1
Emergencia (E)..... = 1 0
Chequeo (CH)..... = 0 1

Y de esta forma tenemos la tabla de verdad:

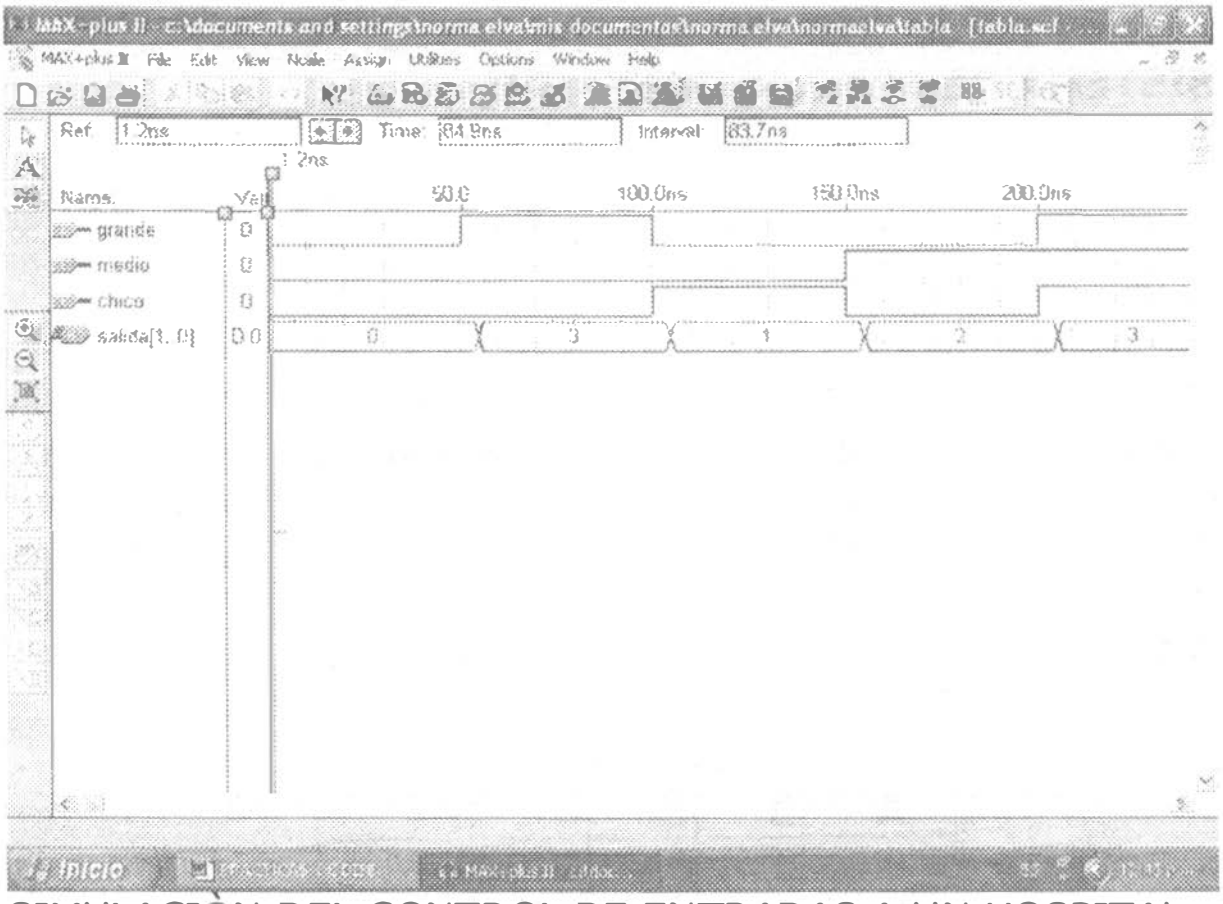
ENTRADAS			SALIDAS
SE	E	CH	Turno
1	*	*	1 1
0	1	*	1 0
0	0	0	0 0
0	0	0	0 0

Por lo tanto introduciendo la tabla al lenguaje Verilog HDL:

```
module tabla(SE,E,CH,TURNO);
input SE,E,CH;
output [1:0] TURNO;
reg [1:0] TURNO;
always
begin
casex ({SE,E,CH})
  3'b1xx  : begin TURNO=2'b11; end
  3'b01x  : begin TURNO=2'b10; end
  3'b001  : begin TURNO=2'b01; end
  3'b000  : begin TURNO=2'b00; end
 endcase
end
endmodule
```



CONTROL DE ENTRADAS A UN HOSPITAL USANDO EL LENGUAJE VERILOGHDL.



SIMULACIÒN DEL CONTROL DE ENTRADAS A UN HOSPITAL UTILIZANDO LENGUAJE VERILOG HDL.

INDICE

SISTEMA DE ALARMA HOTELERO

OBJETIVO:

Familiarizarse con el editor gráfico utilizando elementos SSI.

ESPECIFICACIONES:

Un minihotel tiene 4 habitaciones (A, B, C, D) y dos botones, cada habitación tiene un timbre para servicio y ellos serán atendidos por los botones. Desafortunadamente a los ocupantes de las habitaciones A y B no les gusta el primer botón y no aceptarán sus servicios. Dibuja la tabla de verdad y el circuito correspondiente en donde suene la alarma cuando una requisición no pueda ser atendida.

SOLUCIÓN:

Tabla de verdad

A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$f(A\ B\ C\ D) = \Sigma\ (m_7, m_{11}, m_{12}, m_{13}, m_{14}, m_{15})$

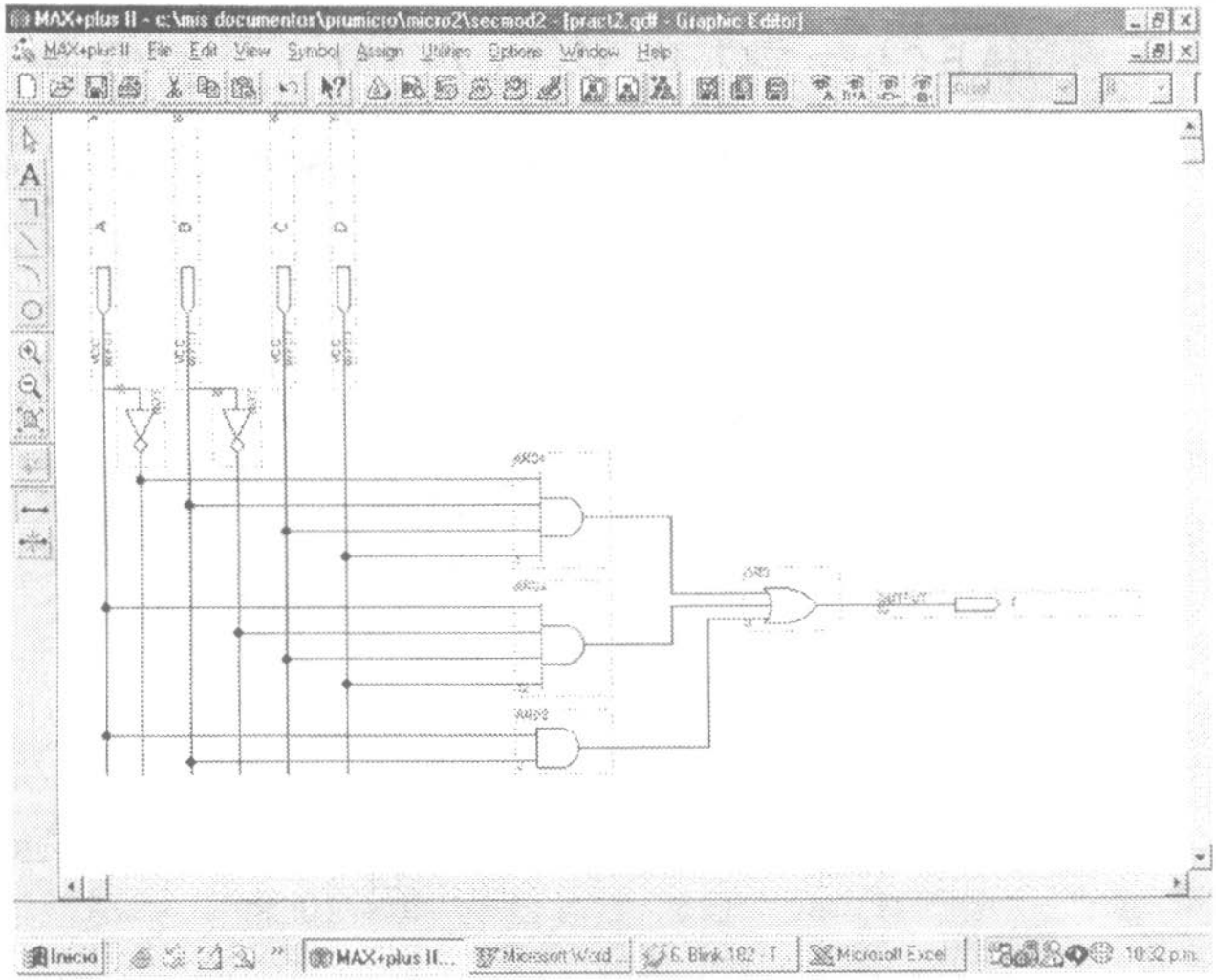
$f(A\ B\ C\ D) = \Sigma\ (7, 11, 12, 13, 14, 15)$

$f = A'BCD + AB'CD + ABC'D' + ABC'D + ABCD' + ABCD$

Minimizando:

$F = A'BCD + AB'CD + AB$

Implementando:



SISTEMA DE ALARMA HOTELERO

INDICE

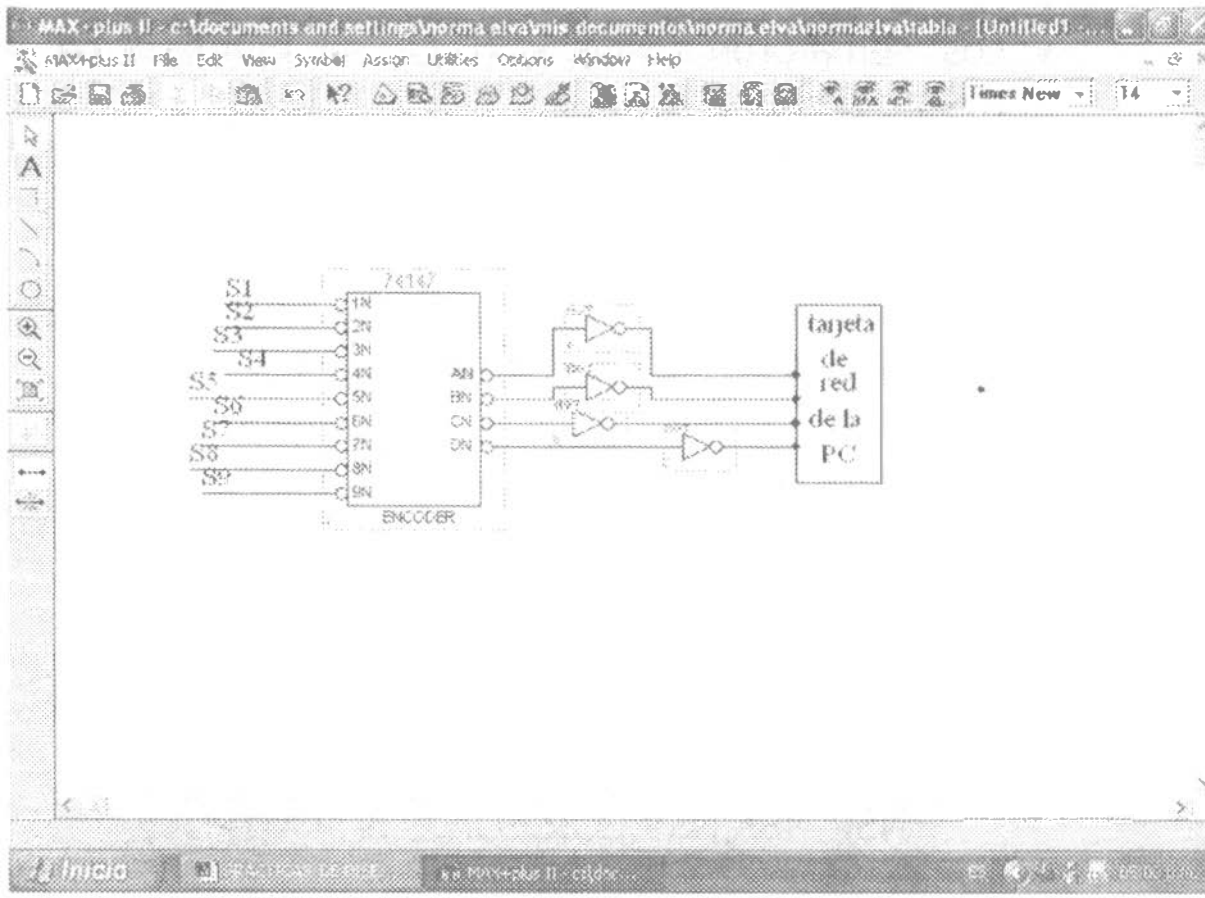
SISTEMA DE SEGURIDAD DE UN BANCO.

OBJETIVO:

Familiarizarse con el editor gráfico utilizando elementos MSI:

ESPECIFICACIONES:

Hacer el controlador de seguridad de un banco en la que estén supervisando 10 puntos distintos y cuando alguien pase por alguno de ellos, el sensor mandará una señal que activará una alarma que a su vez mandará un mensaje vía Internet, notificando que un intruso se encuentra en el banco.



SISTEMA DE SEGURIDAD DE UN BANCO

INDICE

CONTROL DE SEMÁFOROS DEL CRUCE DE DOS AVENIDAS

OBJETIVO:

Hacer el controlador de los semáforos de un cruce de dos avenidas, ambas de dobles sentido, utilizando lenguaje AHDL.

ESPECIFICACIONES:

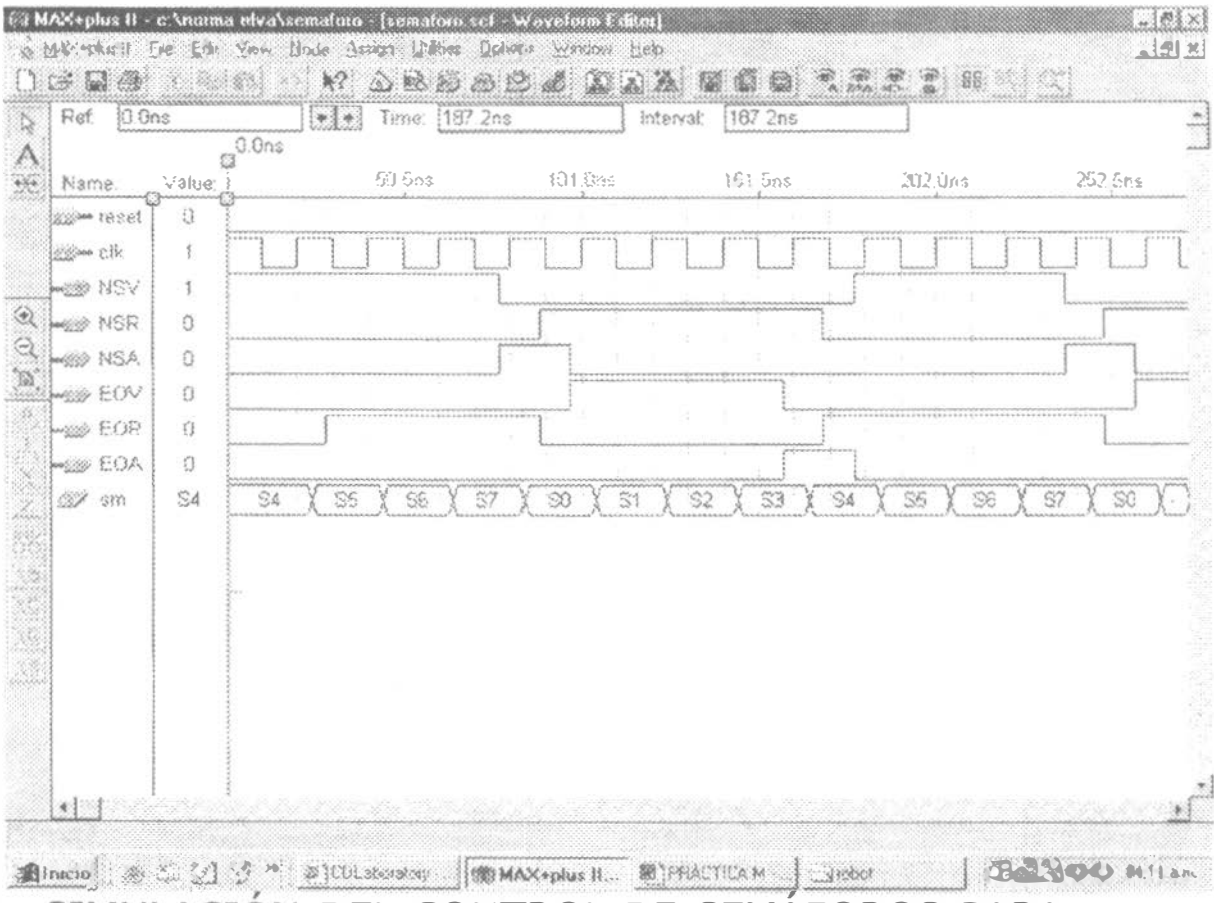
- Habrá cuatro semáforos en cuatro distintas direcciones.
- Los semáforos (N-S) y (E-O) se pondrán en rojo al mismo tiempo
- (N-S) y (E-O) nunca tendrán la luz verde o amarilla al mismo tiempo.
- Las luces verdes durarán 45 segundos.
- Las luces amarillas durarán 15 segundos.
- Las luces rojas durarán 60 segundos.

Introduciendo el diseño:

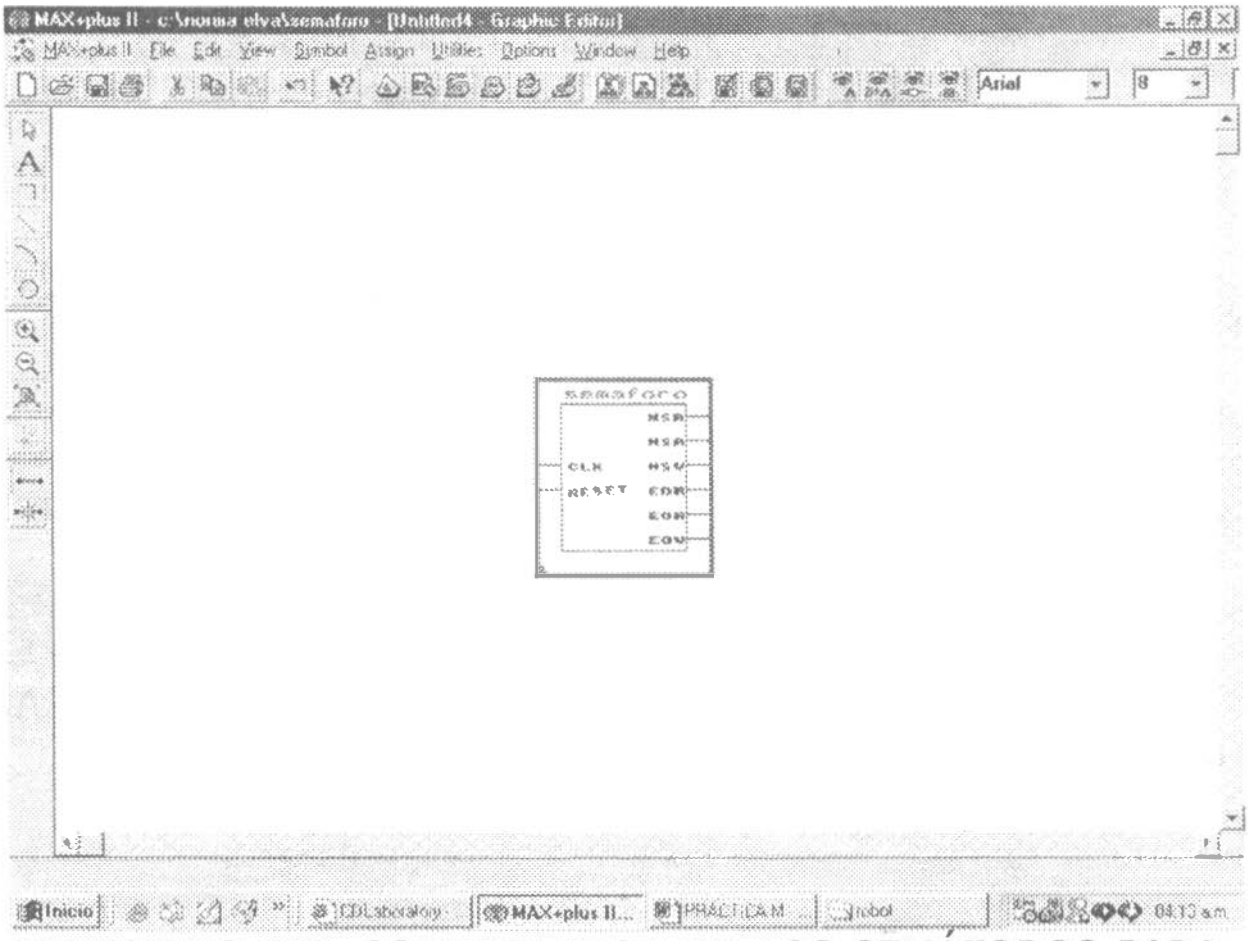
```
TITLE "semaforo";
SUBDESIGN semaforo
(
  clk,reset  : INPUT ;
  NSR,NSA,NSV,EOR,EOA,EOV: OUTPUT;
)
VARIABLE
      sm:MACHINE OF BITS (b2,b1,b0)
```

```
WITH STATES (
     S0=b"000",
     S1=b"001",
     S2=b"010",
     S3=b"011",
     S4=b"100",
          S5=b"101",
     S6=b"110",
     S7=b"111");
BEGIN
DEFAULTS
  NSR=gnd;
   NSA=gnd;
   NSV=gnd;
   EOR=gnd;
   EOA=gnd;
   EOV=gnd;
END DEFAULTS;
sm.clk=clk;
sm.reset=reset;
case sm is
   WHEN S0=>
    NSA=gnd;
    NSR=vcc;
    EOV=vcc;
    EOR=gnd;
    sm=S1;
   WHEN S1=>
    NSR=vcc;
    EOV=vcc;
    sm=S2;
   WHEN S2=>
    NSR=vcc;
```

```
EOV=vcc;
sm=S3;
WHEN S3=>
NSR=vcc;
EOV=gnd;
EOA=vcc;
sm=S4;
WHEN S4=>
NSR=gnd;
EOA=gnd;
NSV=vcc;
EOR=vcc;
sm=S5;


WHEN S5=>
NSV=vcc;
EOR=vcc;
sm=S6;
WHEN S6=>
NSV=vcc;
EOR=vcc;
sm=S7;
WHEN S7=>
NSV=gnd;
EOR=vcc;
NSA=vcc;
sm=S0;
END CASE;
END;
```



SIMULACIÓN DEL CONTROL DE SEMÁFOROS PARA EL CRUCE DE DOS AVENIDAS



PROTOTIPO DEL CONTROLADOR DE LOS SEMÁFOROS PARA EL CRUCE DE DOS AVENIDAS

INDICE

DECODIFICADOR DE BCD A 7 SEGMENTOS

OBJETIVO:

Hacer un decodificador de BCD a 7 segmentos utilizando el lenguaje AHDL

ESPECIFICACIONES:

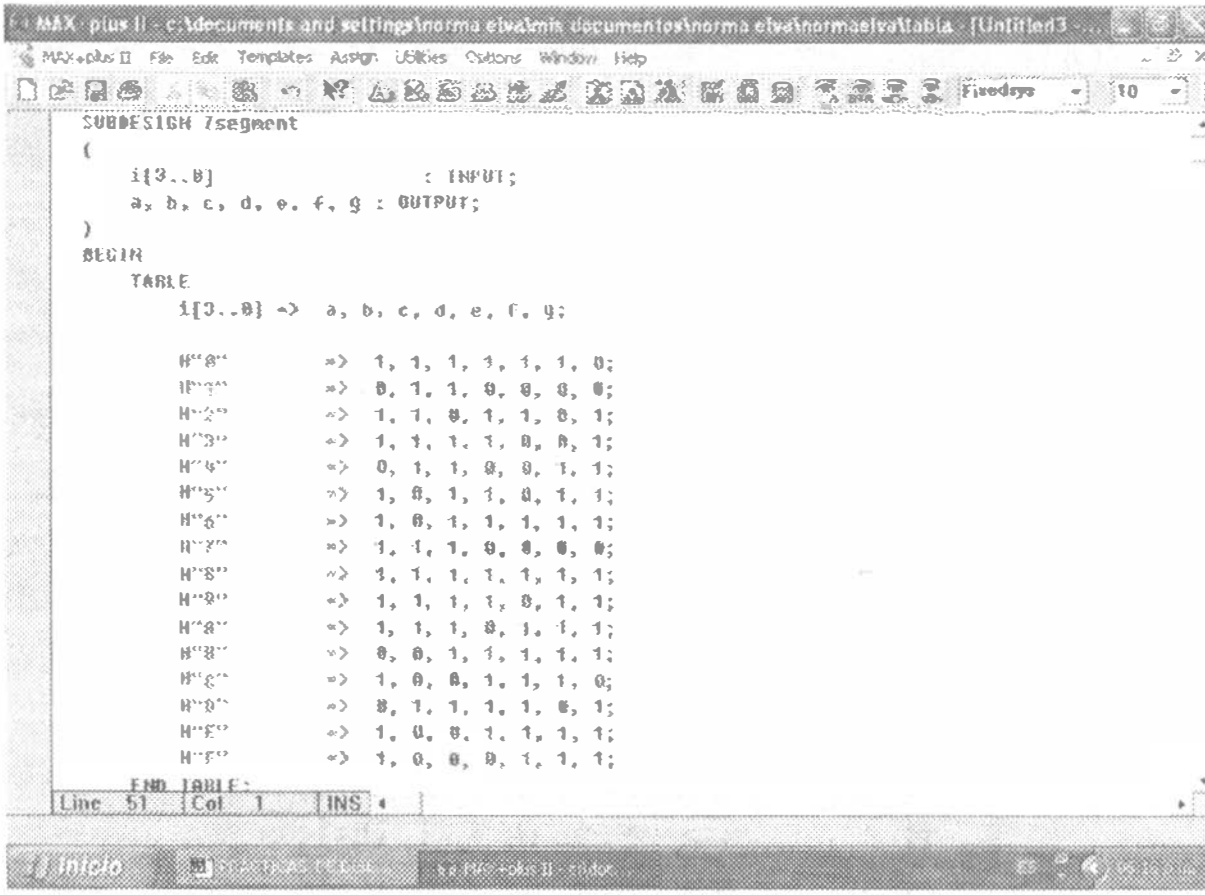
Se tendrá un dipswitch en el cual dependiendo del número decimal que se elija, en un display aparecerá.

SOLUCIÓN:

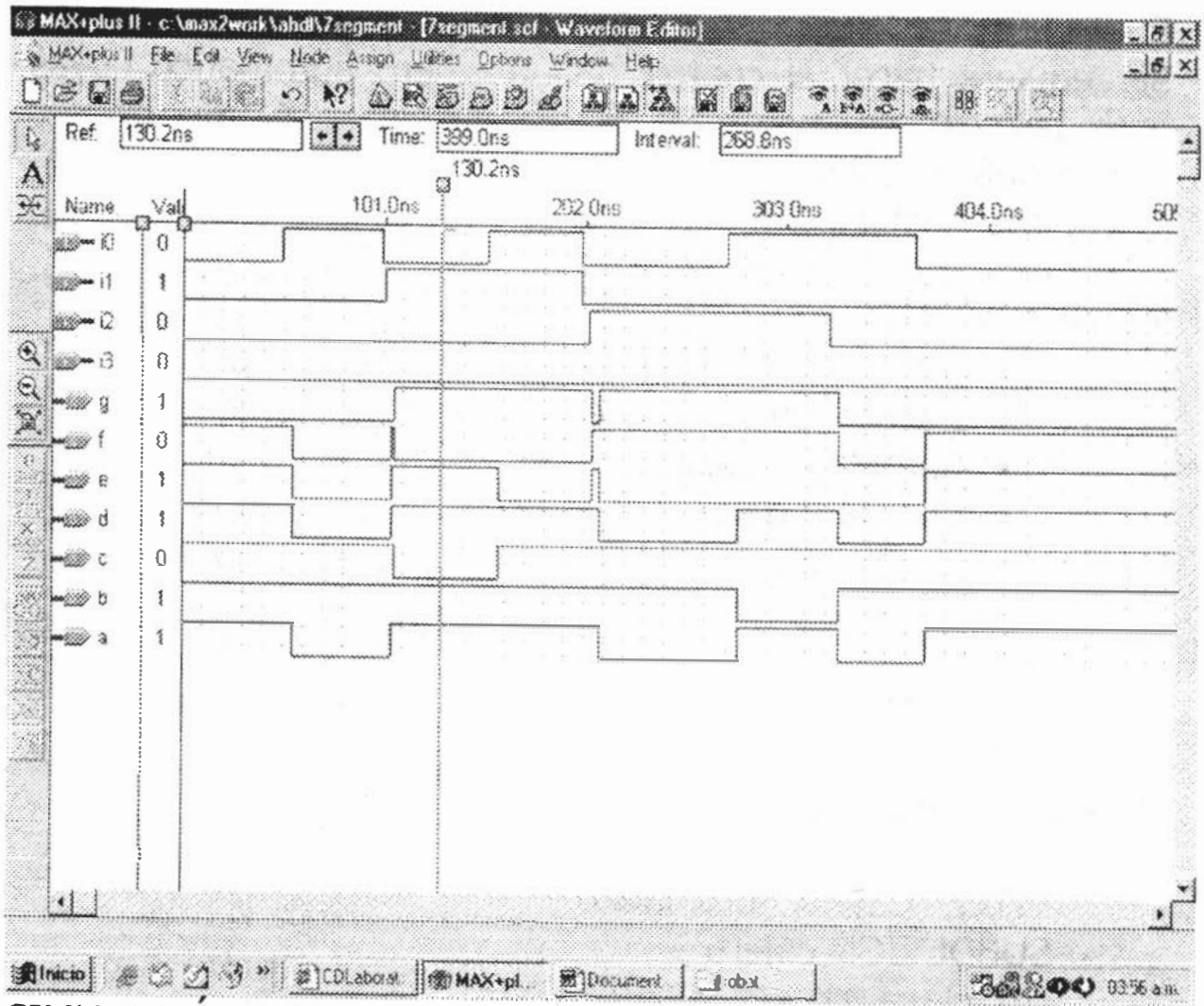
Introduciendo el diseño:

```
SUBDESIGN 7segment
(
    i[3..0]                         : INPUT;
    a, b, c, d, e, f, g     : OUTPUT;
)
BEGIN
    TABLE
        i[3..0]     =>  a, b, c, d, e, f, g;

        H"0"        =>  1, 1, 1, 1, 1, 1, 0;
        H"1"        =>  0, 1, 1, 0, 0, 0, 0;
        H"2"        =>  1, 1, 0, 1, 1, 0, 1;
        H"3"        =>  1, 1, 1, 1, 0, 0, 1;
        H"4"        =>  0, 1, 1, 0, 0, 1, 1;
        H"5"        =>  1, 0, 1, 1, 0, 1, 1;
        H"6"        =>  1, 0, 1, 1, 1, 1, 1;
        H"7"        =>  1, 1, 1, 0, 0, 0, 0;
```

```
        H"8"    =>  1, 1, 1, 1, 1, 1, 1;
        H"9"    =>  1, 1, 1, 1, 0, 1, 1;
        H"A"    =>  1, 1, 1, 0, 1, 1, 1;
        H"B"    =>  0, 0, 1, 1, 1, 1, 1;
        H"C"    =>  1, 0, 0, 1, 1, 1, 0;
        H"D"    =>  0, 1, 1, 1, 1, 0, 1;
        H"E"    =>  1, 0, 0, 1, 1, 1, 1;
        H"F"    =>  1, 0, 0, 0, 1, 1, 1;
    END TABLE;
END;
```



PROGRAMA DE BCD A 7 SEGMENTOS UTILIZANDO AHDL



SIMULACIÓN DEL PROGRAMA DE BCD A 7 SEGMENTOS

INDICE

CONTROL DE UN CAJERO AUTOMÁTICO

OBJETIVO:

Hacer el controlador de un cajero automático, utilizando el lenguaje AHDL.

ESPECIFICACIONES:

Se requiere hacer el control de un cajero automático cuyas entradas sean :

- tarjeta recibida = tp.
- Clave recibida = cl.
- Comparación correcta = cc.

Y cuyas salidas sean :

- Introduzca su tarjeta = smpl.
- Introduzca su clave = smcl.
- Cantidad deseada = mca.
- Compara datos con el sistema = cmpc.
- Limpia el comparador = limp.
- Entrega dinero = ed

SOLUCIÓN:

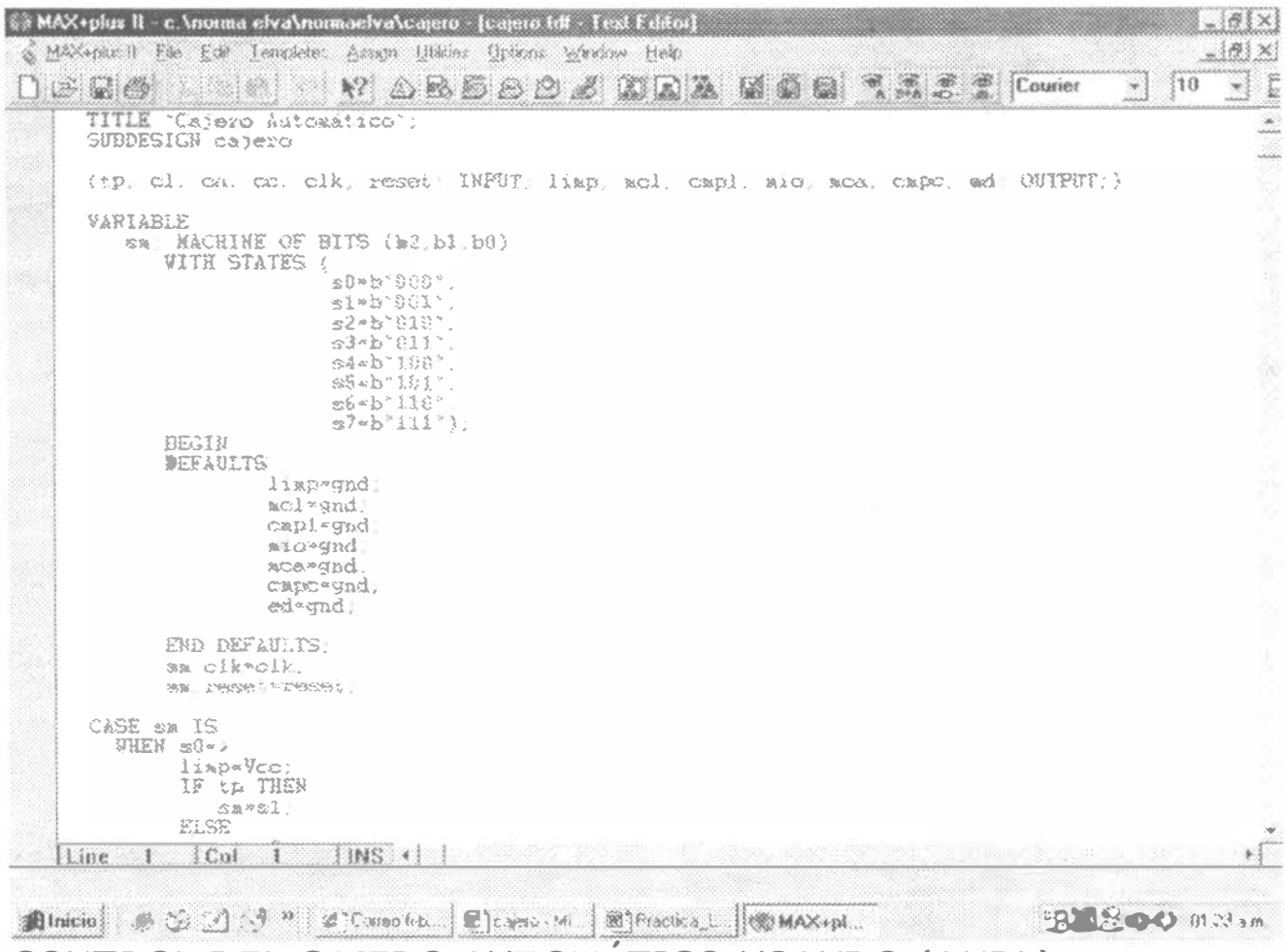
Usando el lenguaje AHDL:

```
TITLE "Cajero Automático";
SUBDESIGN cajero
(tp, cl, ca, cc, clk, reset: INPUT;
 limp,smcl, cmpl, mio, mca, cmpc, ed: OUTPUT;)
VARIABLE
   sm: MACHINE OF BITS (b2,b1,b0)
```

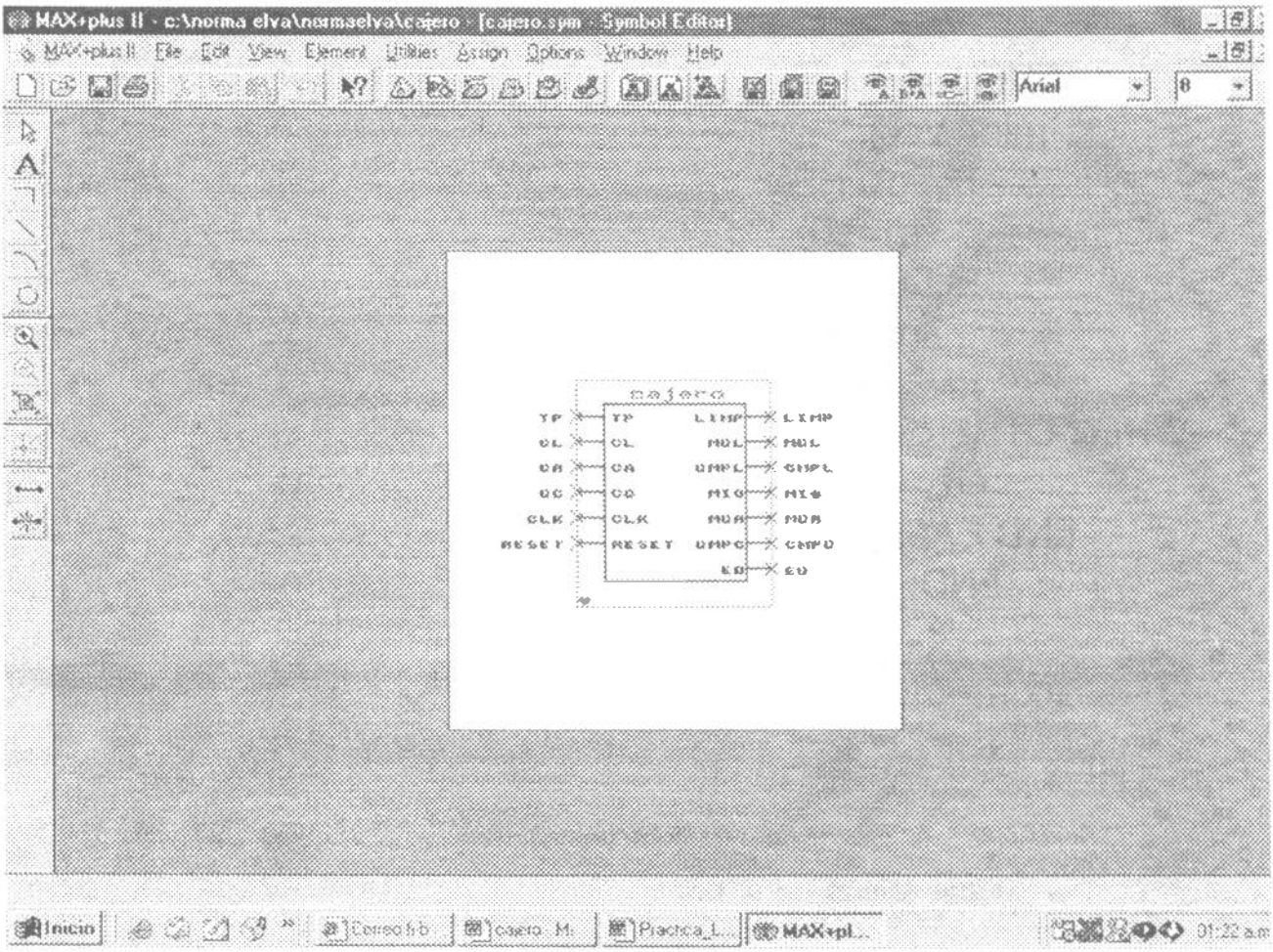
```
    WITH STATES (
              s0=b"000",
              s1=b"001",
              s2=b"010",
              s3=b"011",
            s4=b"100",
              s5=b"101",
              s6=b"110",
             s7=b"111");
    BEGIN
    DEFAULTS
          limp=gnd;
          mcl=gnd;
          cmpl=gnd;
          mio=gnd;
          mca=gnd;
          cmpc=gnd;
          ed=gnd;
END DEFAULTS;
    sm.clk=clk;
    sm.reset=reset;

CASE sm IS
  WHEN s0=>
       limp=Vcc;
      IF tp THEN
        sm=s1;
      ELSE
        sm=s3;
      END IF;
  WHEN s1=>
```

```
mcl=Vcc;
        sm=s2;

       WHEN s2=>
        cmpl=Vcc;
            IF cl THEN
              sm=s4;
            ELSE
              sm=s3;
            END IF;

       WHEN s3=>
        mio=Vcc;
            IF ca THEN
              sm=s0;

            ELSE
              sm=s1;
            END IF;

       WHEN s4=>
        mca=Vcc;
        sm=s5;
WHEN s5=>
        cmpc=Vcc;
            IF cc THEN
              sm=s7;
            ELSE
              sm=s6;
            END IF;
```

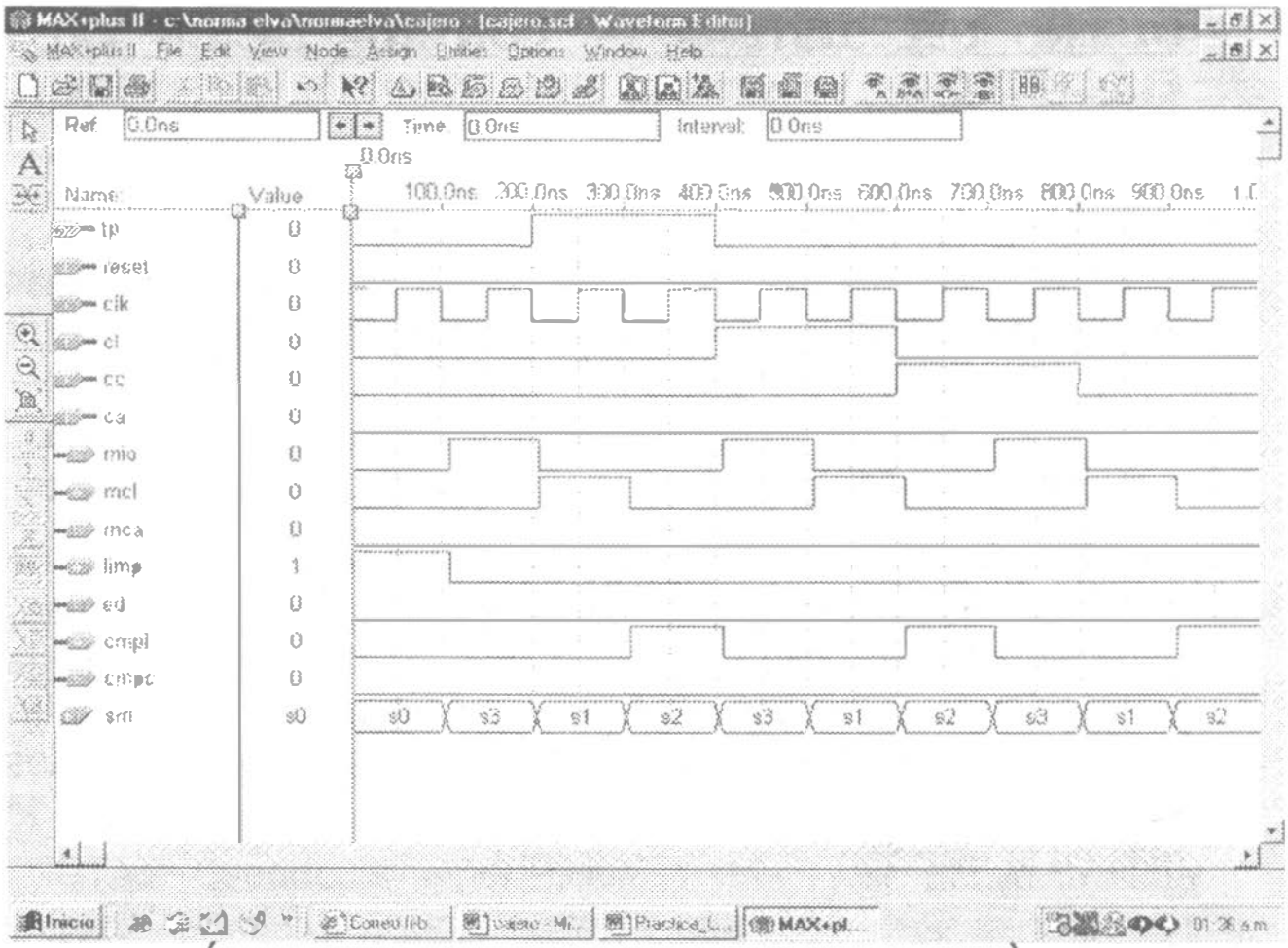
```
        WHEN s6=>
          mio=Vcc;
              IF ca THEN
                 sm=s0;
              ELSE
                 sm=s4;
              END IF;
WHEN OTHERS=>
              ed=vcc;
        END CASE;
            END;
```



CONTROL DEL CAJERO AUTOMÁTICO USANDO (AHDL)



PROTOTIPO DEL CONTROLADOR DEL CAJERO AUTOMÁTICO



SIMULACIÓN DEL CONTROL DE UN CAJERO AUTOMÀTICO

INDICE

DISEÑO DEL SISTEMA DE CONTROL DE UN TREN ELÉCTRICO

OBJETIVO:

Aplicar los conceptos de Cartas ASM para la solución de problemas y desarrollar un circuito capaz de poder realizar la simulación del funcionamiento de un tren.

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 n 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.

Solución:

Los sensores que detectan la presencia del tren en cada estación, están conectados a una compuerta OR. La salida de esta "OR" es llamada "estación" e indica con un uno cuando el tren entra en cada estación, sin importar a cual entra. Solo interesa saber cuando entra a las estaciones terminales (SA y SN), para cambiar de dirección se tiene que mandar movimiento del tren. Se tiene un módulo que dada la señal de activación alto2 se genera una señal de salida de 2 minutos, por otra parte, se tiene otro módulo que generará una señal de 1 minuto cuando una emergencia ocurra. Esta señal es

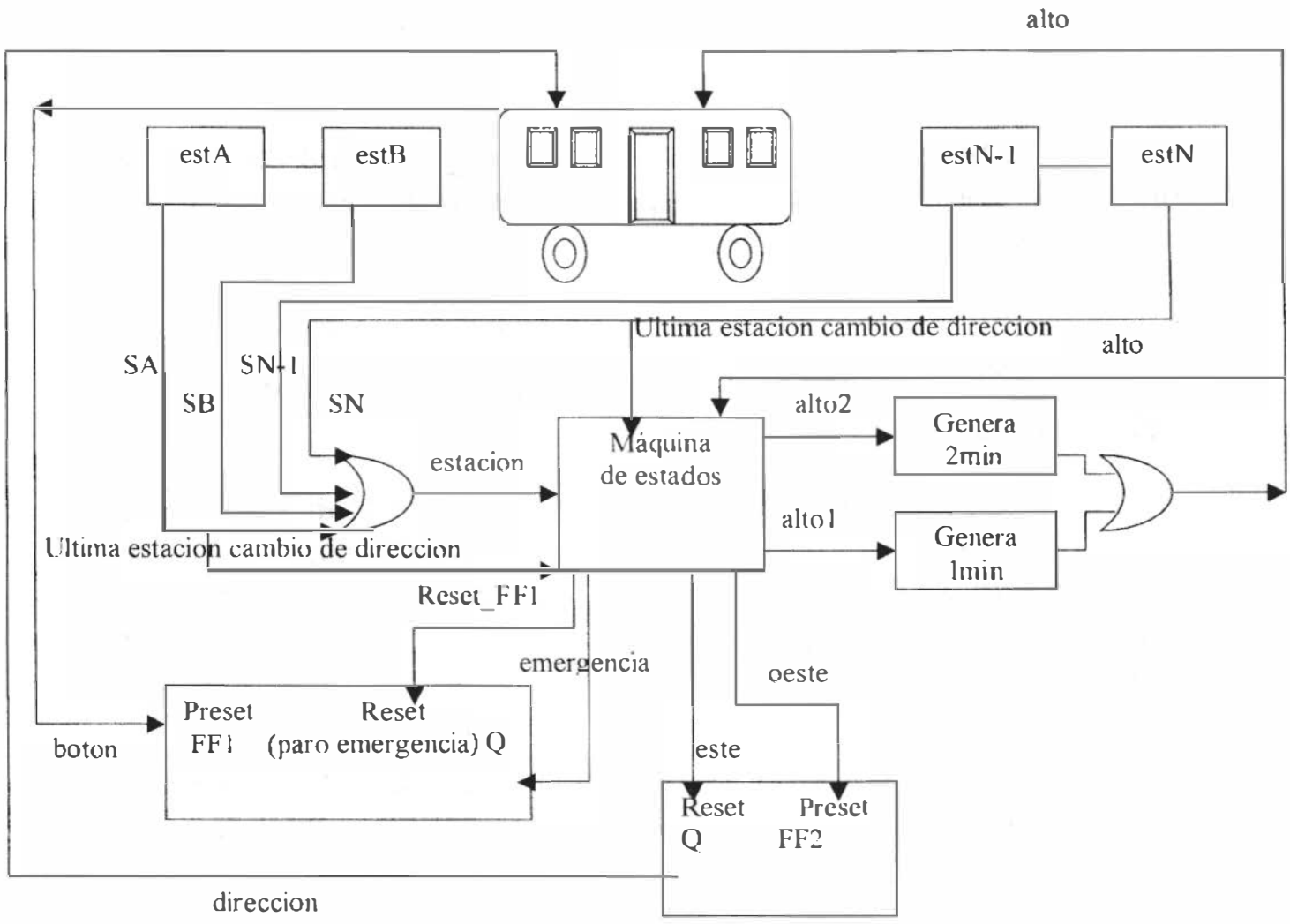
activada con alto1. Estas 2 señales son entradas a una "OR" cuya salida detendrá al tren ya sea por 2 o por 1 minuto extra, según sea el caso. Además la señal "alto" se retroalimenta a la máquina de estados.

Se tiene un flip-flop el cual será puesto en uno cuando se oprima el botón de emergencia en el tren. Esto se logra conectando la línea del "botón" en el preset del flip-flop. La salida de este flip-flop entra a la máquina de estados, como "emergencia". Este flip-flop tiene que ser puesto a cero para aceptar otra emergencia y esto se hace por medio de la línea reset del flip-flop.

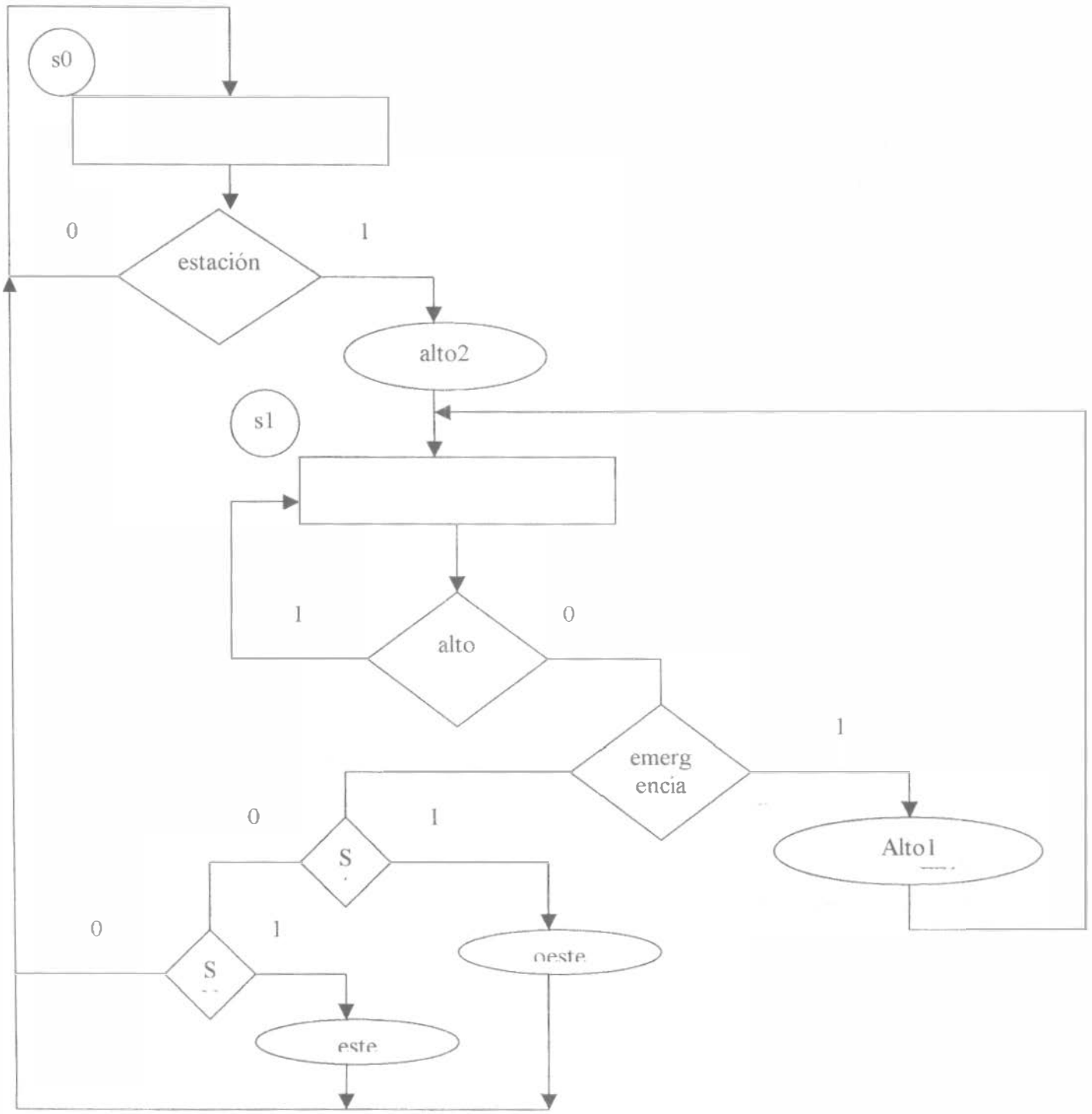
Hay también un flip-flop de direcciones el cual le indica al tren la dirección a seguir, si la salida del flip-flop es igual a cero, el tren ira hacia el oeste y si es uno irá hacia el este. El tren se está moviendo todo el tiempo a menos que la señal de "alto" este activada.

NOTA: Cuando la señal estación vale 1, se activa la señal alto2, la cual hará que se genere una señal de 2 minutos que parará al tren. En el estado est1 se rehúsa la variable alto y si ésta vale 1, entonces se queda esperando en ese estado hasta que valga cero.

El diagrama del sistema digital es el siguiente:



La carta ASM de la máquina de estados es la siguiente:

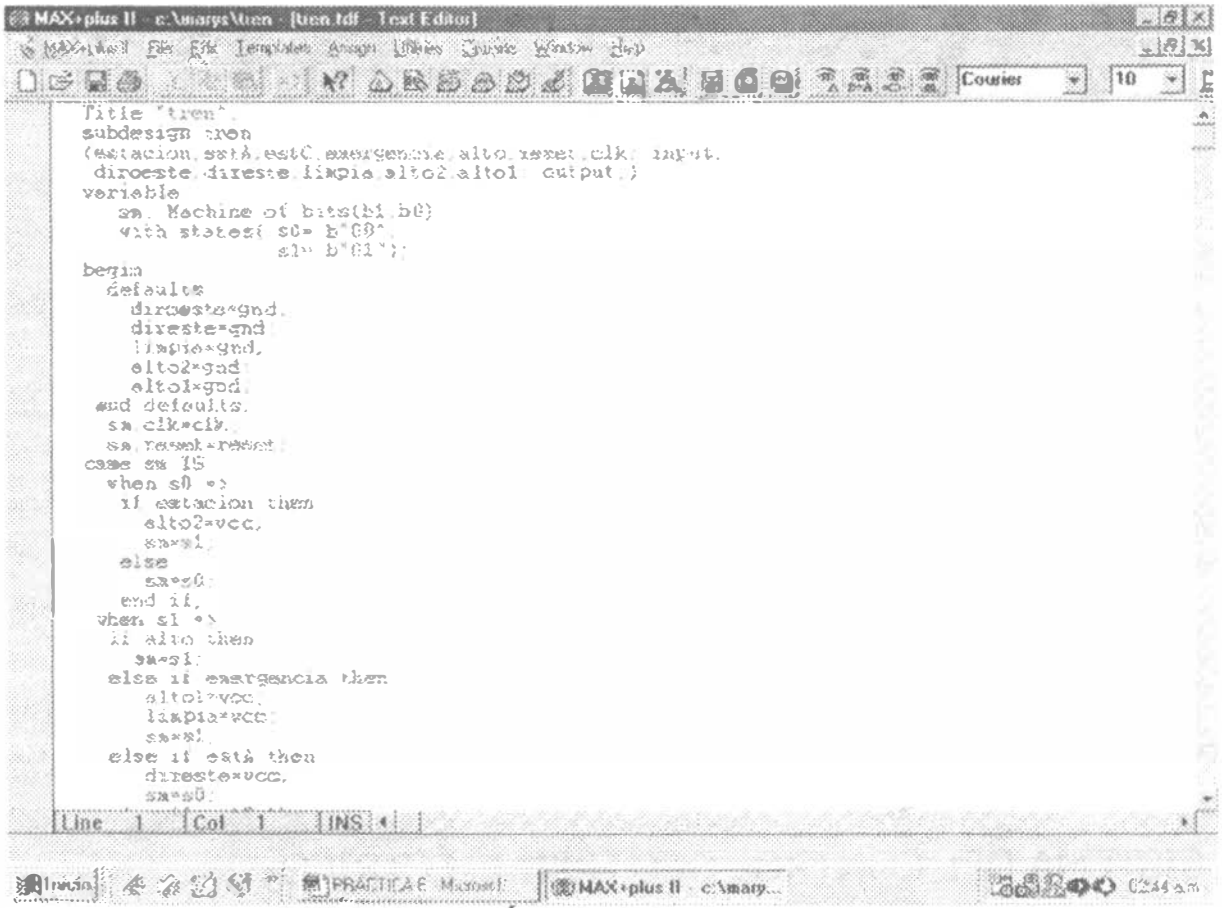


El código es el siguiente:

- Para el módulo del tren

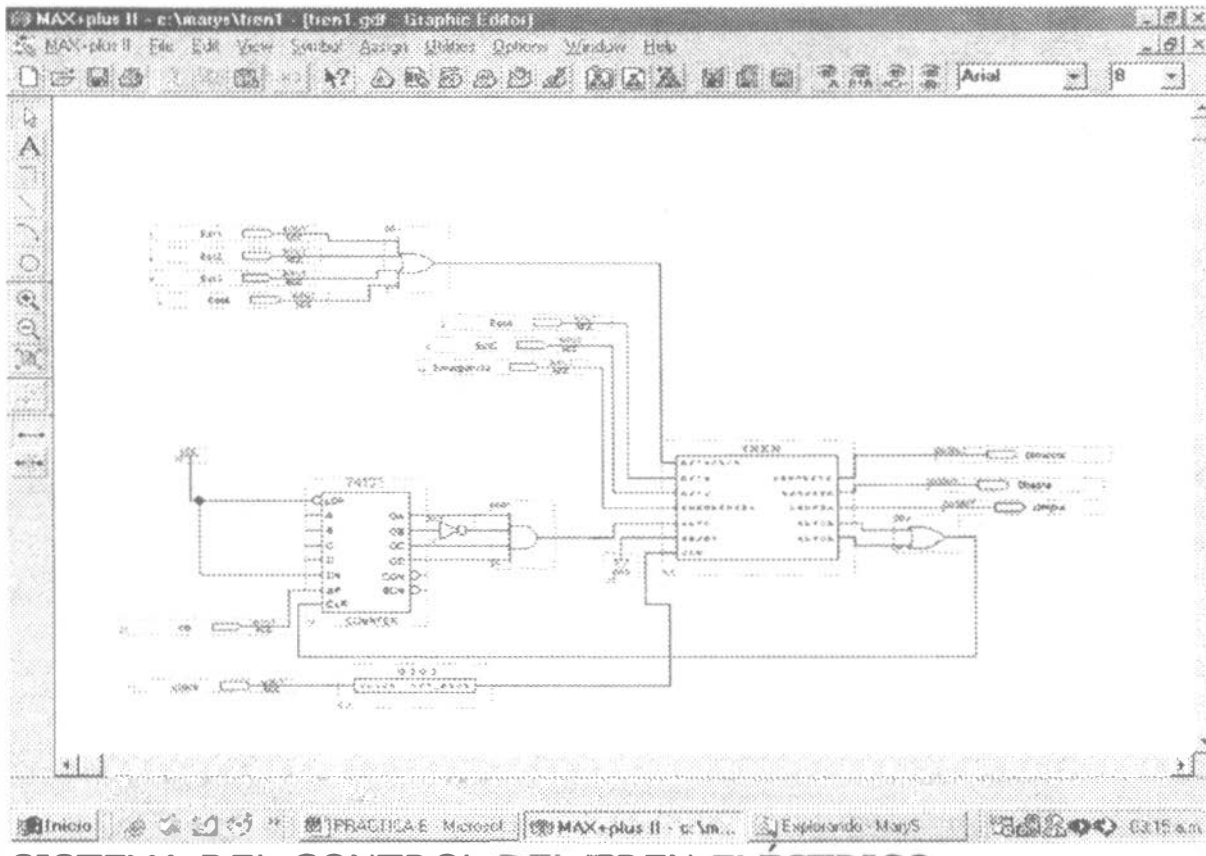
```
Title "tren";
subdesign tren
(estacion,estA,estC,emergencia,alto,reset,clk: input;

diroeste,direste,limpia,alto2,alto1: output;)
variable
  sm:        Machine        of bits(b1,b0)
   with states( s0= b"00",
                s1= b"01");
begin
  defaults
       diroeste=gnd;
       direste=gnd;
       limpia=gnd;
       alto2=gnd;
       alto1=gnd;
 end defaults;
  sm.clk=clk;
  sm.reset=reset;
case sm IS
  when s0 =>
   if estacion then
     alto2=vcc;
     sm=s1;
   else
     sm=s0;
   end if;
 when s1 =>
  if alto then
    sm=s1;
  else if emergencia then
     alto1=vcc;
     limpia=vcc;
     sm=s1;
  else if estA then
     direste=vcc;
     sm=s0;
  else if estC then
     diroeste=vcc;
     sm=s0;
   else
     sm=s0;
   end if;
  end if;
 end if;
end if;
end case;
end;
```

Dentro del editor de texto se coloca el código anterior:



CONTROL DE UN TRE ELÉCTRICO USANDO AHDL

Posteriormente se hace en el gràfico todo el diseño que cuenta con el control y periféricos.



SISTEMA DEL CONTROL DEL TREN ELÉCTRICO.

INDICE

DIVISOR DE TIEMPO:

OBJETIVO:

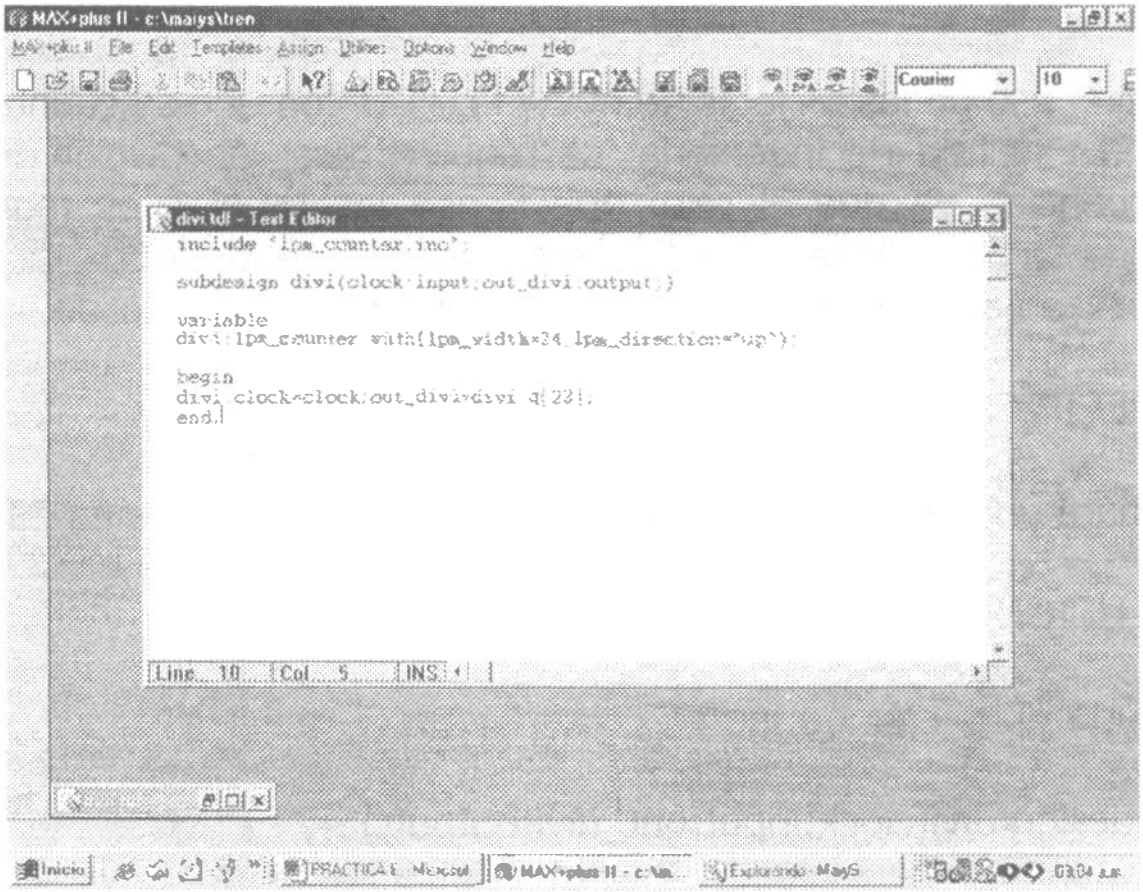
Para el uso de las tarjetas de MAX+PLUS II con que se cuenta en la Facultad de ingeniería, se requiere usar un divisor de frecuencia ya que el reloj que tienen las tarjetas es muy rápida su frecuencia y por lo tanto muy corto el período y por esa razón a los ojos humanos no se ve cambio en la salida.

ESPECIFICACIONES:

Se hará un divisor que tenga 24 líneas, y se usará la línea 23.

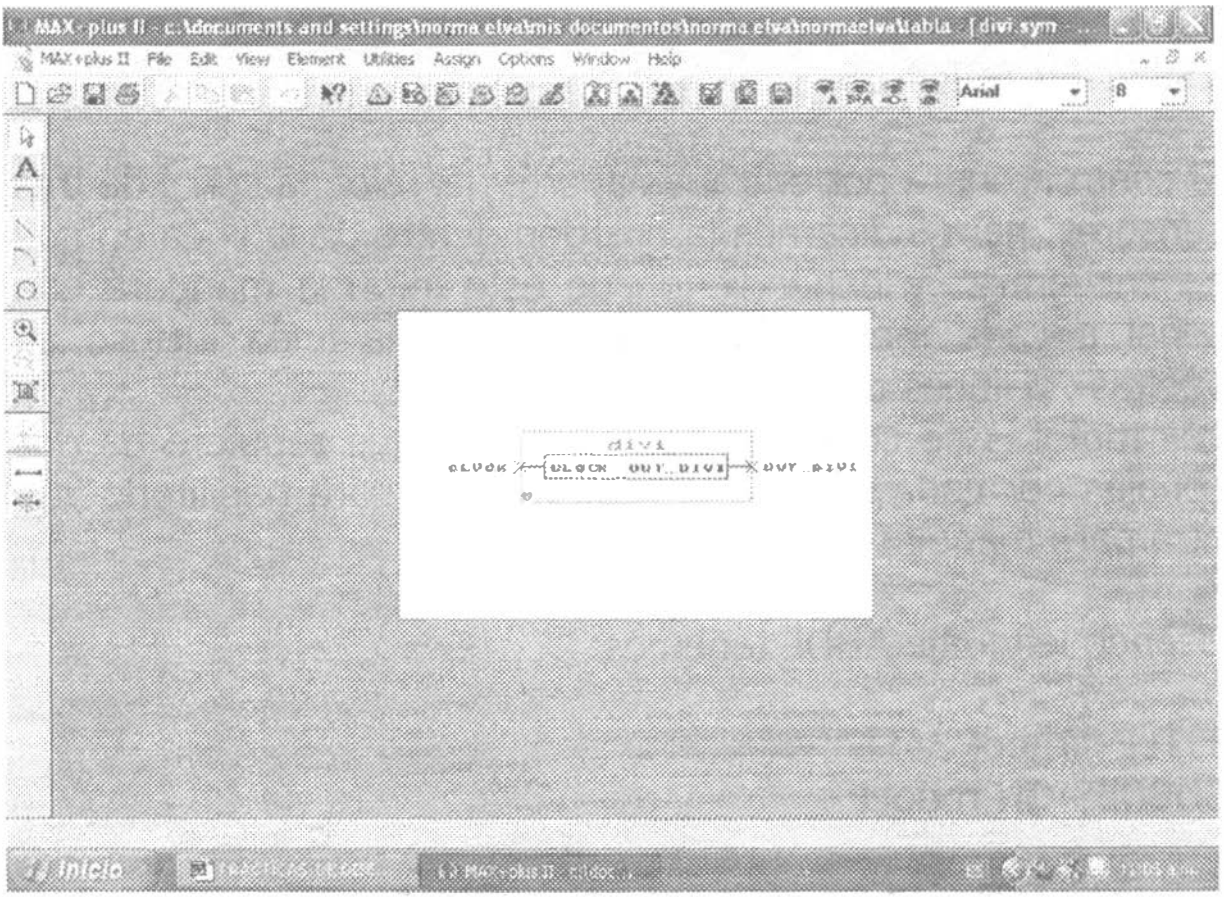
SOLUCIÓN:

```
include "lpm_counter.inc";

subdesign divi(clock:input;out_divi:output;)

variable
divi:lpm_counter
with(lpm_width=24,lpm_direction="up");

begin
divi.clock=clock;out_divi=divi.q[23];
end;
```



DIVISOR DE TIEMPO UTILIZANDO LENGUAJE AHDL

Para poderlo utilizar en cualquier diseño, se requiere tenerlo Como un módulo, por lo que se requiere hacerlo símbolo:



PROTOTIPO DEL DIVISOR DE FRECUENCIA.

INDICE

CONTROL DE UN ROBOT MÓVIL

OBJETIVO:

El Control deberá hacer que el robot móvil siga una línea blanca con fondo negro, utilizando el lenguaje AHDL.

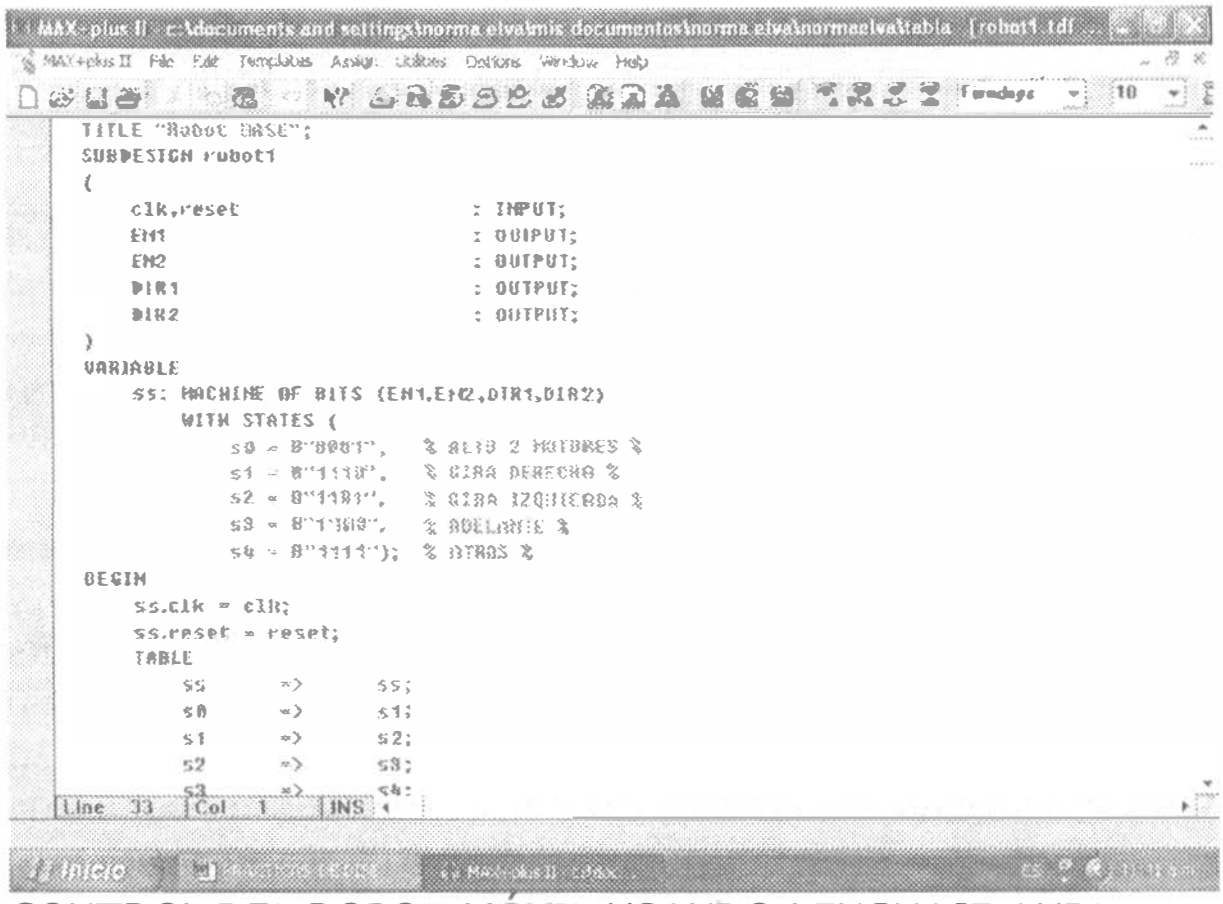
ESPECIFICACIONES:

El control tendrá por entradas aparte del reloj, la señal de tres sensores, para indicarnos la posición dentro de la línea en que se encuentra y dependiendo de éste informe mandará una señal a cada motor para poder seguir la línea incluso en vueltas , o líneas discontinuas. `
Los motores se controlan con un driver que requiere de dos señales por cada motor , una de ellas es para habilitarlo y la otra para manejar el sentido .

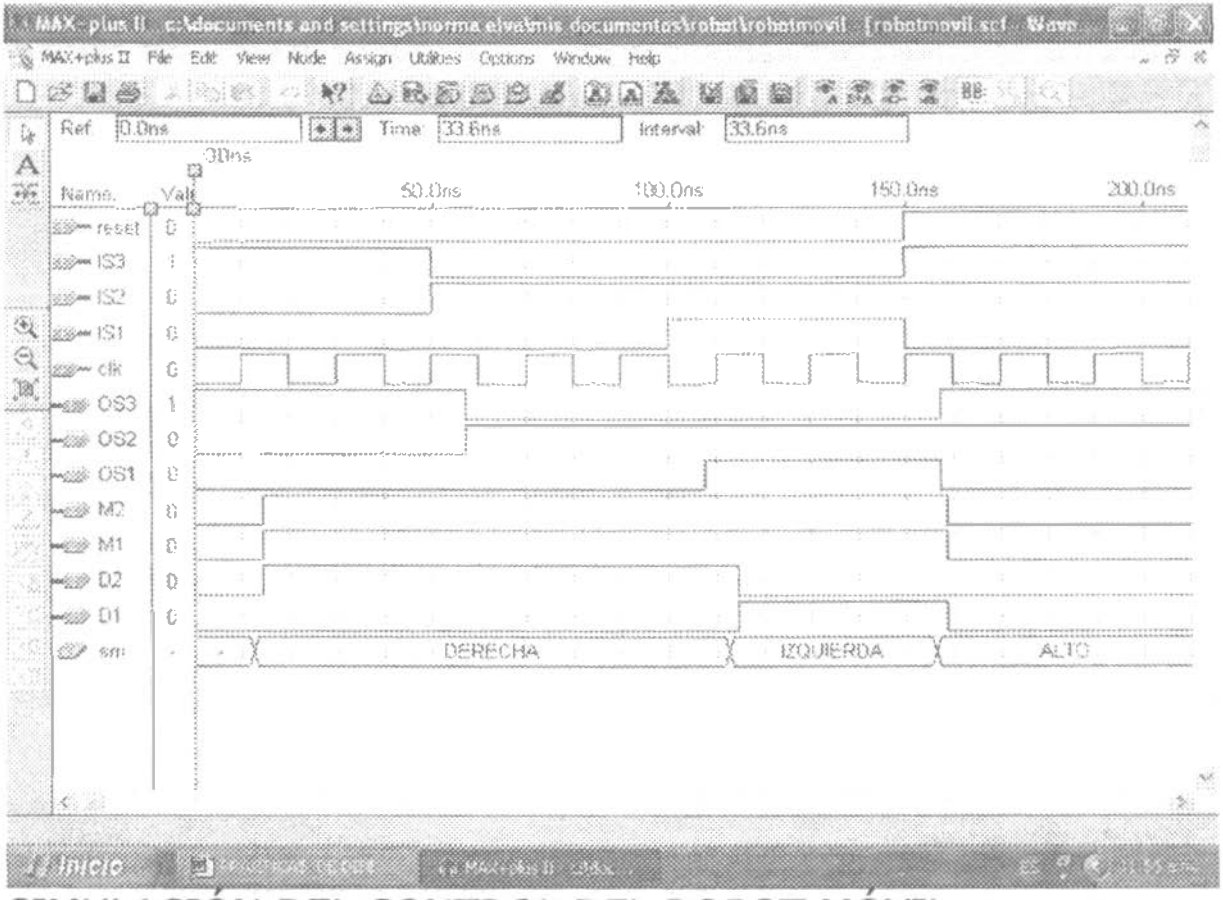
Usando lenguaje AHDL tenemos:

```
TITLE "Robot BASE";
SUBDESIGN robot1
(
      clk,reset                        : INPUT;
      EN1                                    : OUTPUT;
      EN2                                    : OUTPUT;
      DIR1                                   : OUTPUT;
      DIR2                                   : OUTPUT;
)
VARIABLE
      ss: MACHINE OF BITS (EN1,EN2,DIR1,DIR2)
            WITH STATES (
                  s0 = B"0001", % ALTO 2 MOTORES %
```

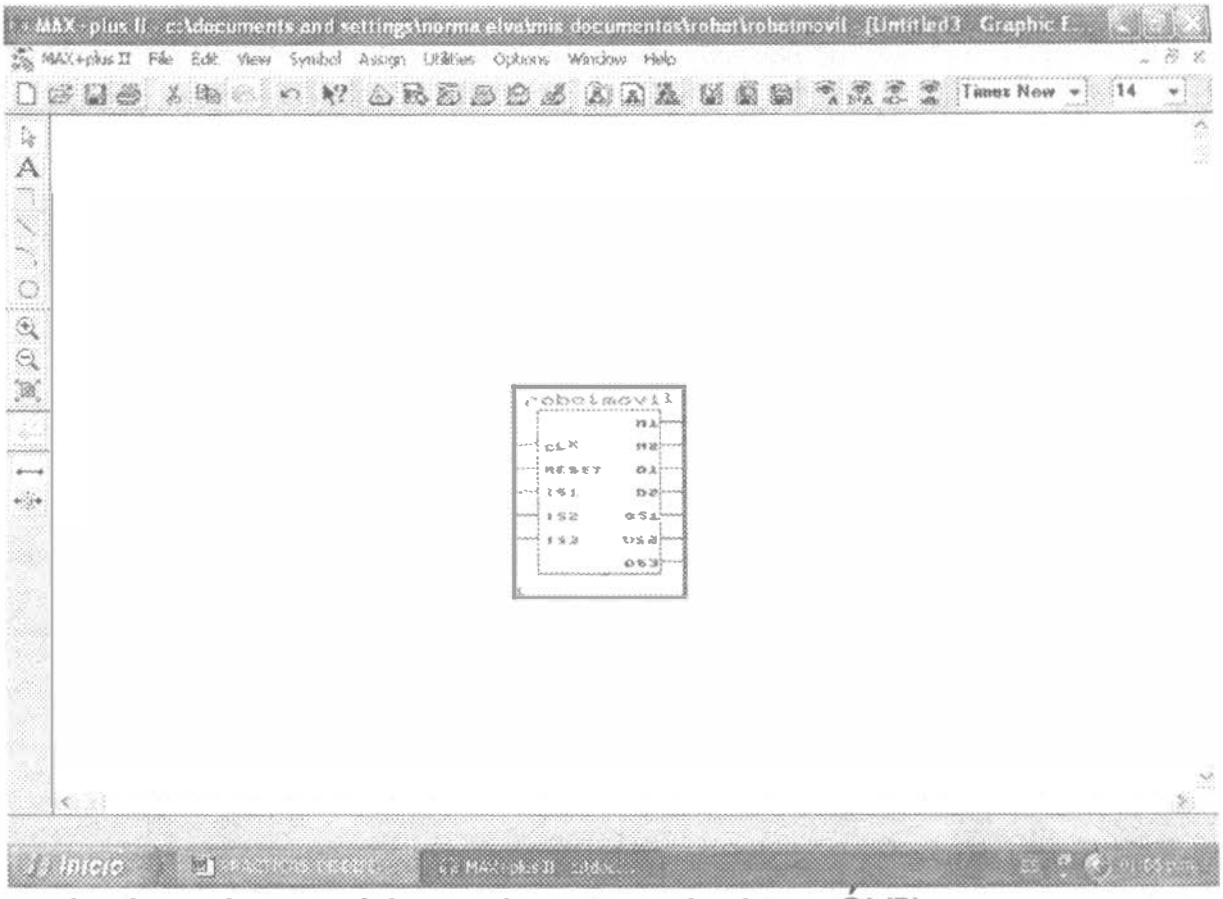
```
                s1 = B"1110", % GIRA DERECHA %
                s2 = B"1101", % GIRA IZQUIERDA %
                s3 = B"1100", % ADELANTE %
                s4 = B"1111");      % ATRAS %
BEGIN
    ss.clk = clk;
    ss.reset = reset;
    TABLE
        ss          =>      ss;
        s0          =>      s1;
        s1          =>      s2;
        s2          =>      s3;
        s3          =>      s4;
        s4          =>      s0;
    END TABLE;
END;
```



CONTROL DEL ROBOT MÓVIL USANDO LENGUAJE AHDL



SIMULACIÓN DEL CONTROL DEL ROBOT MÓVIL.



PROTOTIPO DEL CONTROL DEL ROBOT MÓVIL.

INDICE

DECODIFICADOR DE BCD A 7 SEGMENTOS

OBJETIVO:

Diseñar un decodificador BCD utilizando el lenguaje AHDL.

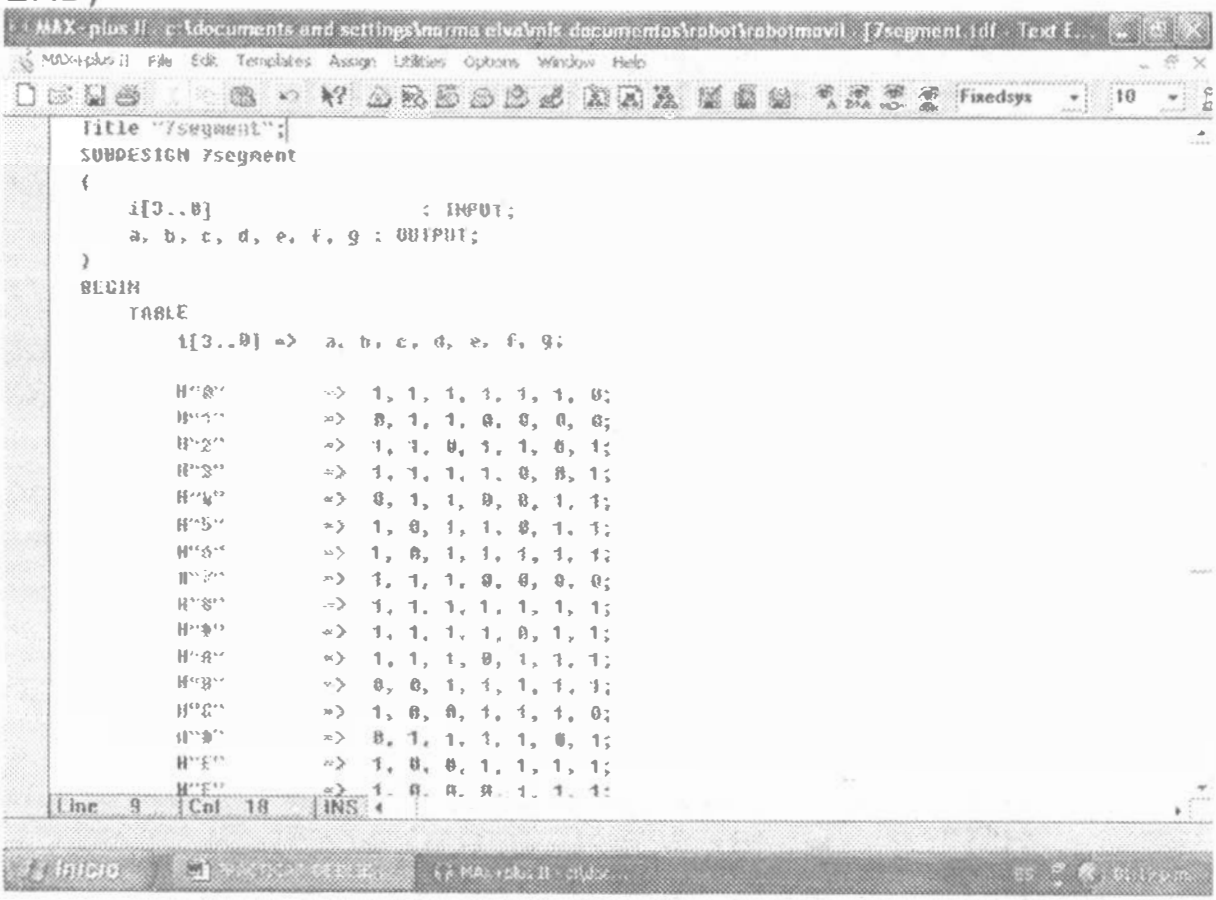
ESPECIFICACIONES :

Se podrá decodificar cualquier número binario de tres bits apara verlo en un display de siete segmentos.

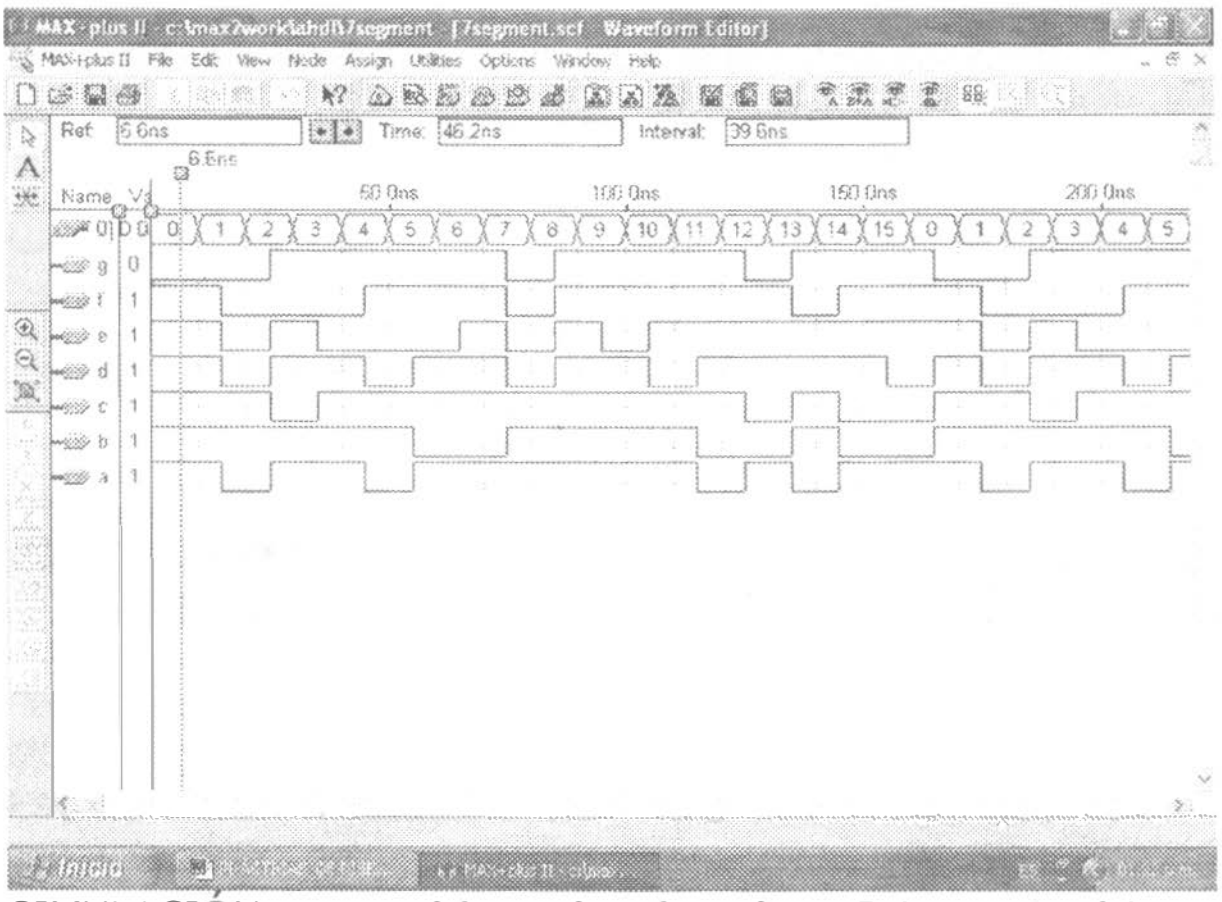
SOLUCIÓN:
Utilizando el lenguaje AHDL:

```
Title "7segment";
SUBDESIGN 7segment
(
    i[3..0]                    : INPUT;
    a, b, c, d, e, f, g        : OUTPUT;
)
BEGIN
    TABLE
        i[3..0]    =>  a, b, c, d, e, f, g;

        H"0"       =>  1, 1, 1, 1, 1, 1, 0;
        H"1"       =>  0, 1, 1, 0, 0, 0, 0;
        H"2"       =>  1, 1, 0, 1, 1, 0, 1;
        H"3"       =>  1, 1, 1, 1, 0, 0, 1;
        H"4"       =>  0, 1, 1, 0, 0, 1, 1;
        H"5"       =>  1, 0, 1, 1, 0, 1, 1;
        H"6"       =>  1, 0, 1, 1, 1, 1, 1;
        H"7"       =>  1, 1, 1, 0, 0, 0, 0;
        H"8"       =>  1, 1, 1, 1, 1, 1, 1;
        H"9"       =>  1, 1, 1, 1, 0, 1, 1;
```

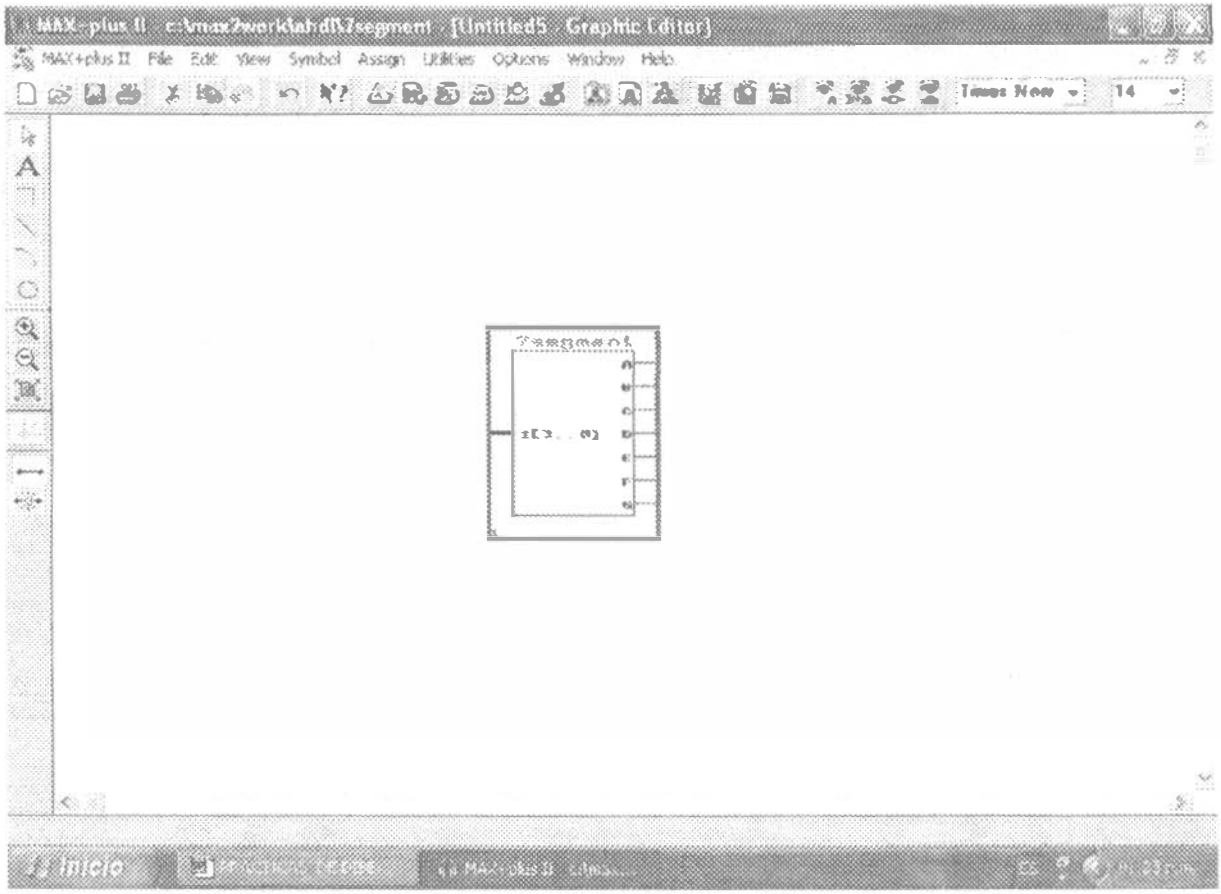
```
        H"A"    =>  1, 1, 1, 0, 1, 1, 1;
        H"B"    =>  0, 0, 1, 1, 1, 1, 1;
        H"C"    =>  1, 0, 0, 1, 1, 1, 0;
        H"D"    =>  0, 1, 1, 1, 1, 0, 1;
        H"E"    =>  1, 0, 0, 1, 1, 1, 1;
        H"F"    =>  1, 0, 0, 0, 1, 1, 1;
    END TABLE;
END;
```



DECODIFICADOR DE BCD A 7 SEGMENTOS UTILIZANDO EL LENGUAJE AHDL.



SIMULACIÓN DEL DECODIFICADOR BCD A 7 SEGMENTOS.



PROTOTIPO DEL DECODIFICADOR BCD A 7SEGMENTOS

INDICE

DISEÑO DE UNA CALCULADORA

OBJETIVO:

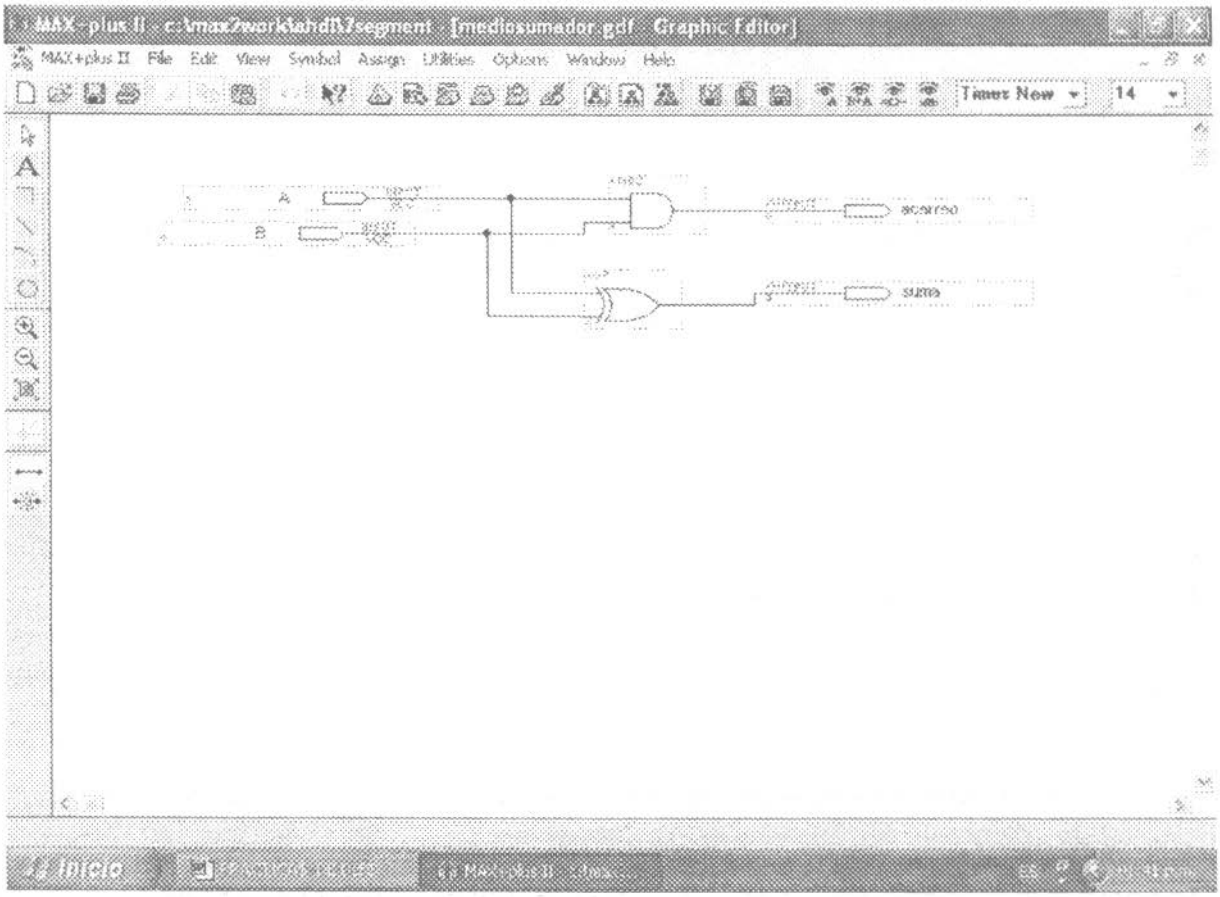
Hacer Una Calculadora , Que Sume, Reste Y Multiplique Utilizando El Editor Gráfico.

ESPECIFICACIONES:

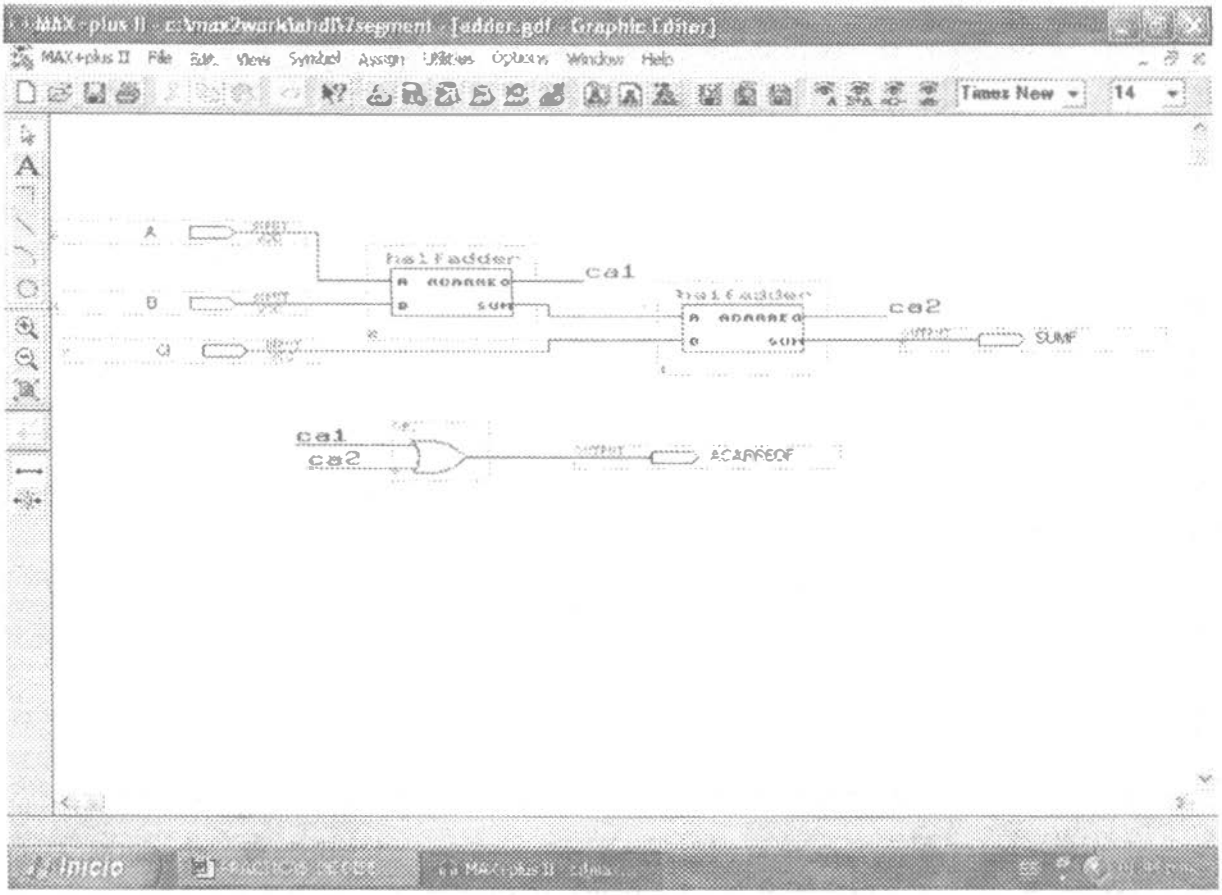
Las operaciones se hará únicamente con números binarios de cuatro bits y se dará el resultado también en binario.

SOLUCIÓN:

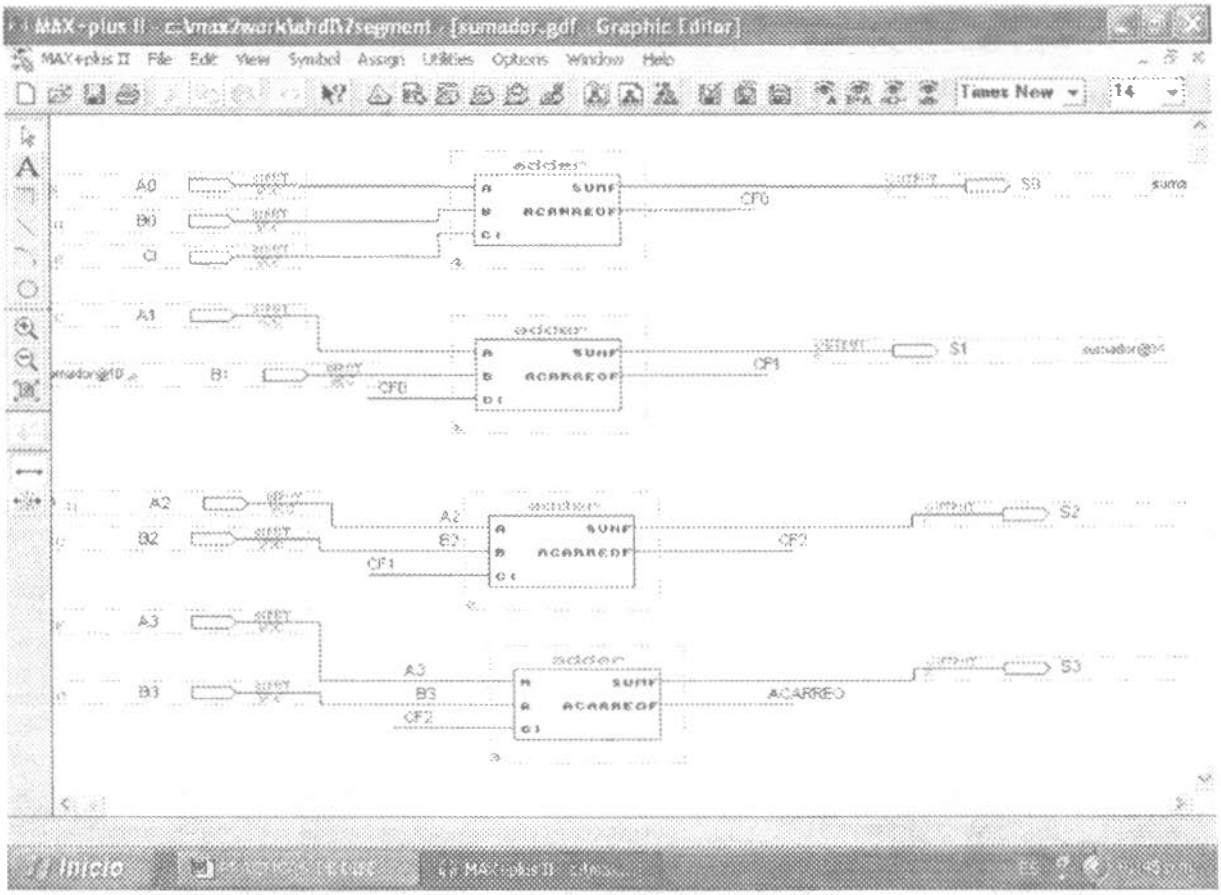
Se hará utilizando subbloques :
Primero el de un medio sumador,
Después se hará el de un sumador completo utilizando dos bloques de medios sumadores,
Posteriormente se hará el sumador de cuatro bits con cuatro bloques de sumadores completos ,
Para hacer un restador – sumador a una de las entradas de cada bloque del sumador completo se le agrega una compuerta exor.
Por otro lado se hace el bloque de la multiplicación y al final con un busmux se unen el bloque de la multiplicación y el de la suma resta.



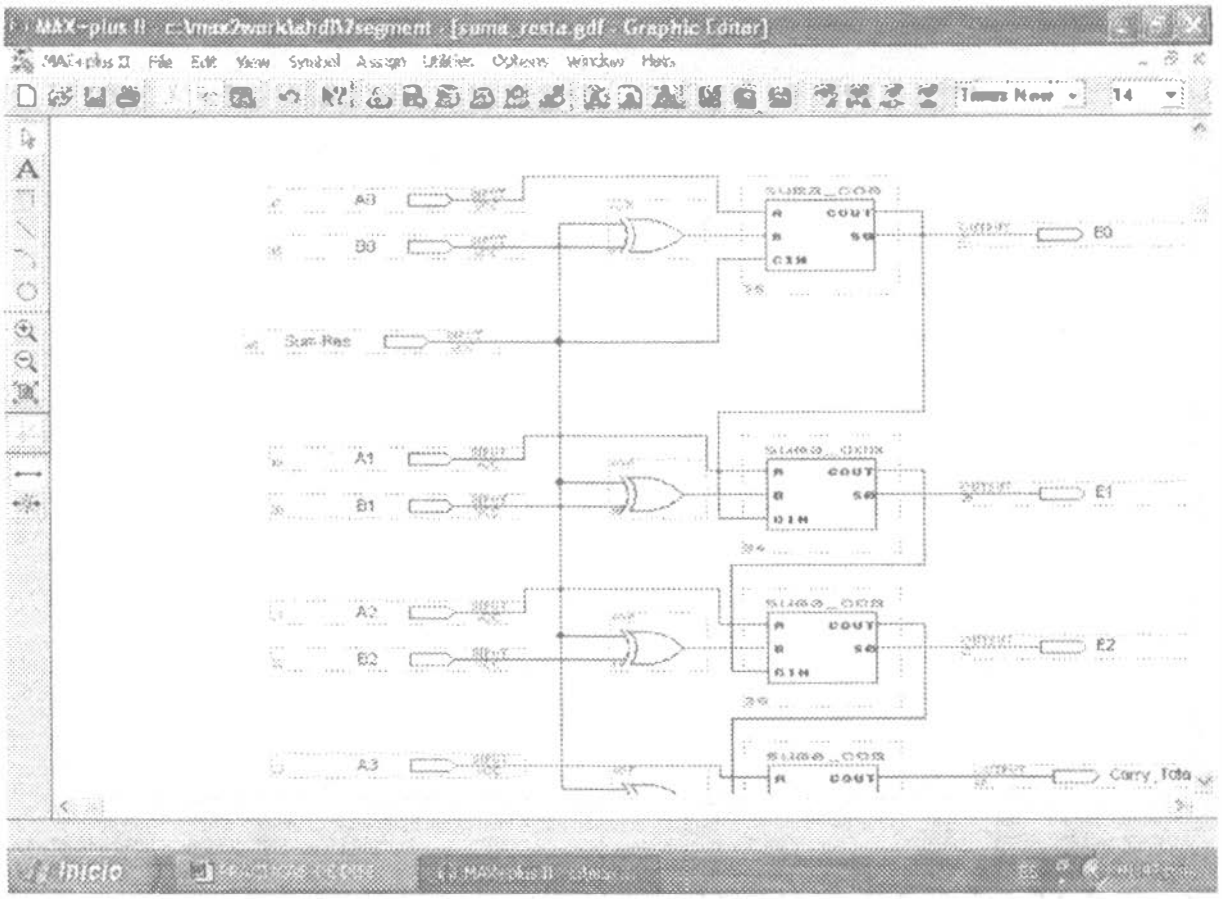
BLOQUE DEL MEDIO SUMADOR



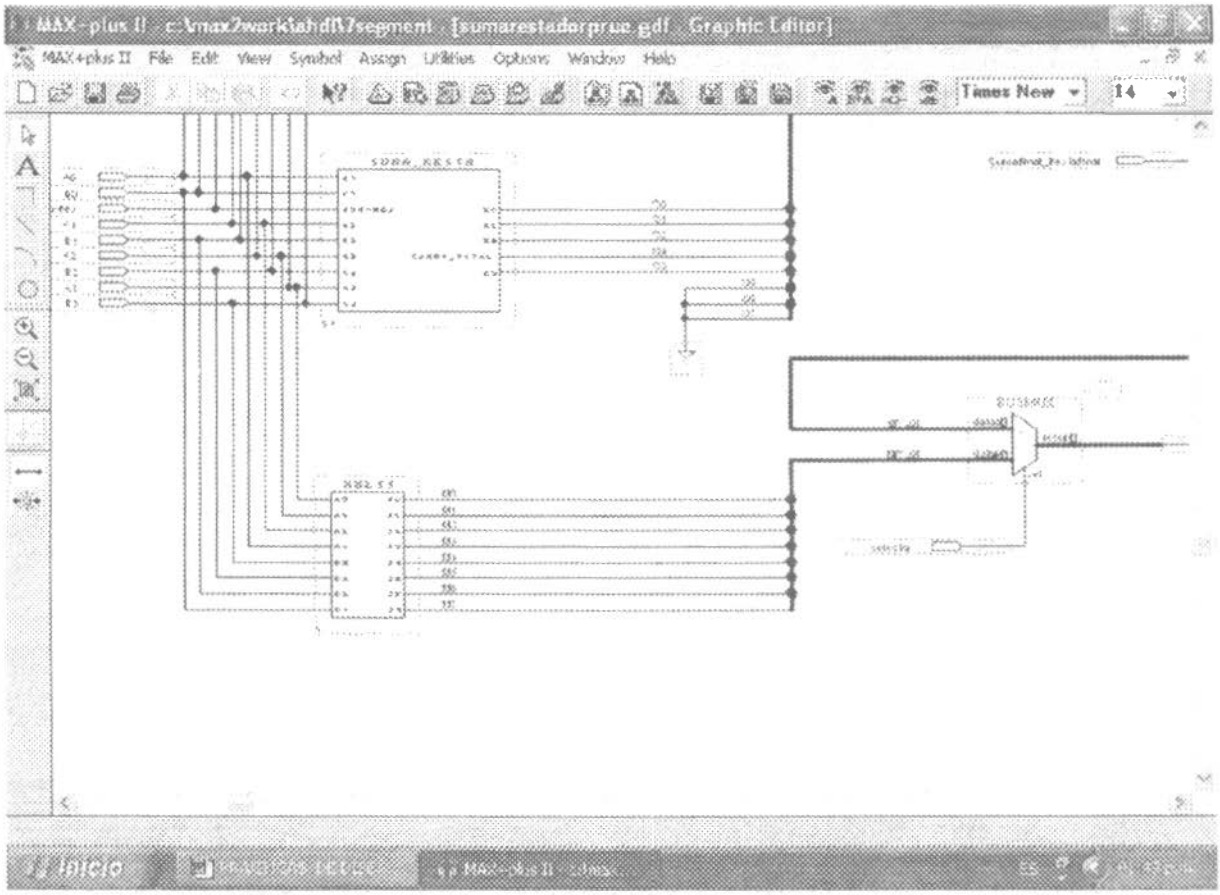
BLOQUE DEL SUMADOR COMPLETO



BLOQUE DEL SUMADOR DE CUATRO BITS.



BLOQUE DEL SUMADOR-RESTADOR DE CUATRO BITS



CALCULADORA DE CUATRO BITS.

INDICE

DISEÑO DE UNA CHAPA ELECTRÓNICA

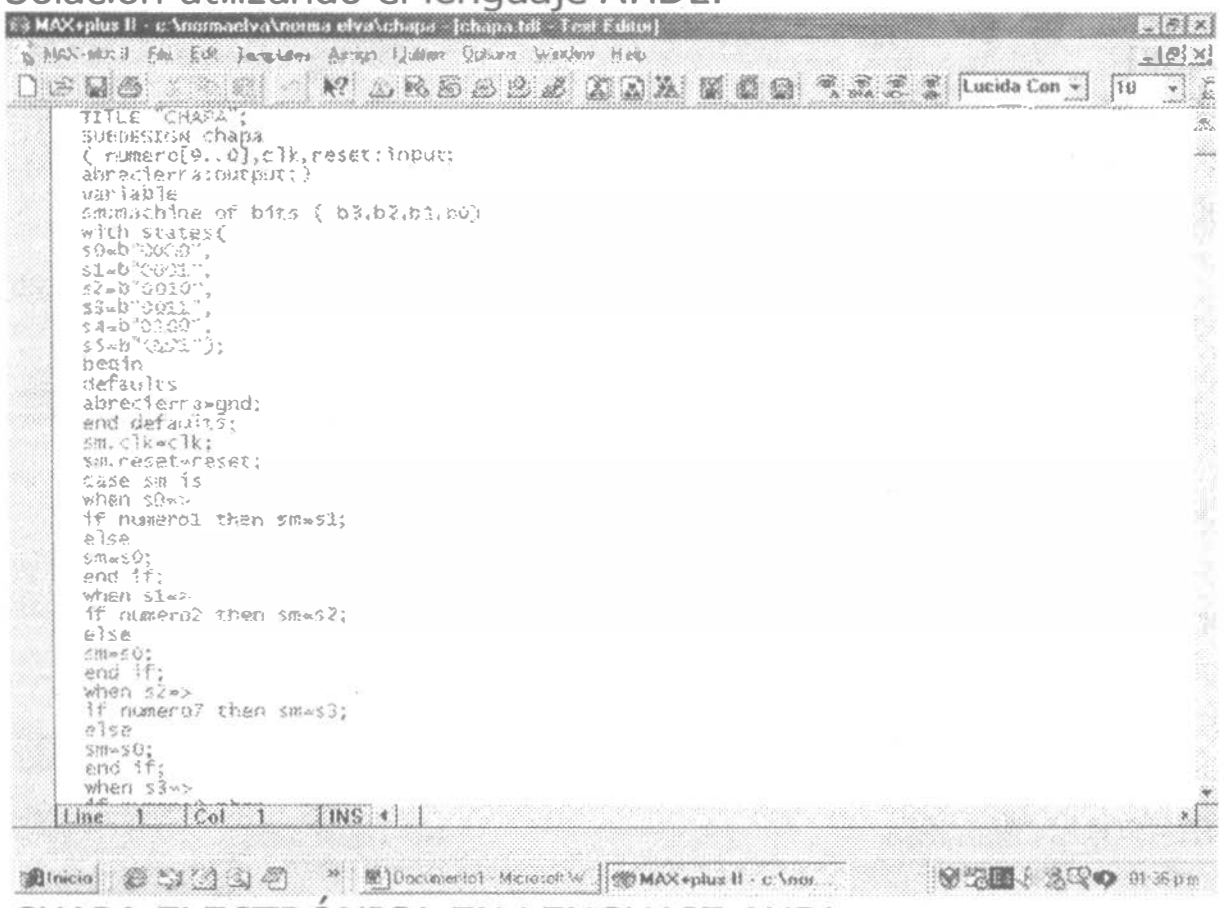
OBJETIVO:
Diseñar el control de una cerradura , utilizando el lenguaje AHDL.

ESPECIFICACIONES :

La cerradura se abre cuando se teclea los números 1,2,7,9.

Solución:

Solución utilizando el lenguaje AHDL:

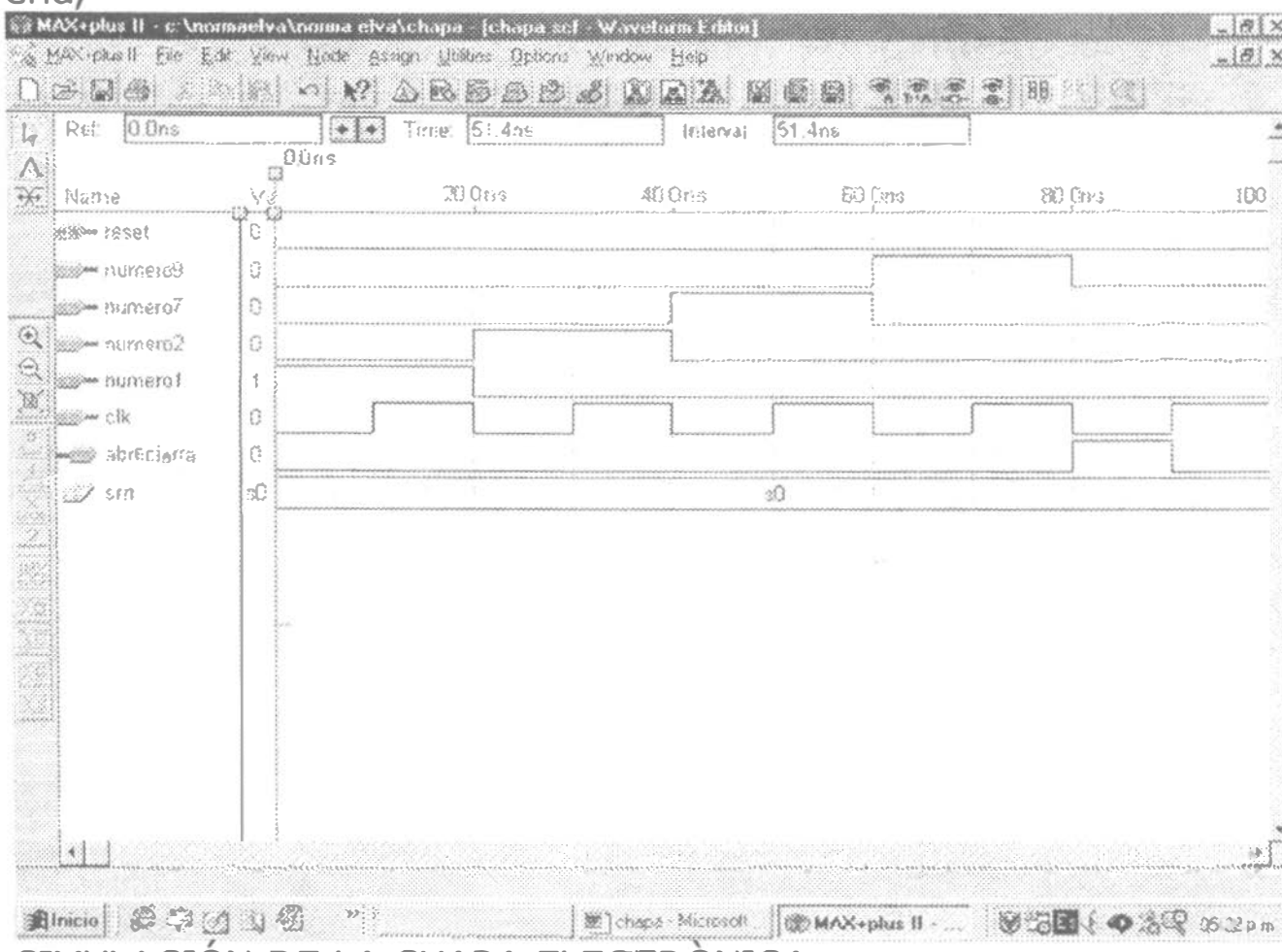


CHAPA ELECTRÓNICA EN LENGUAJE AHDL.

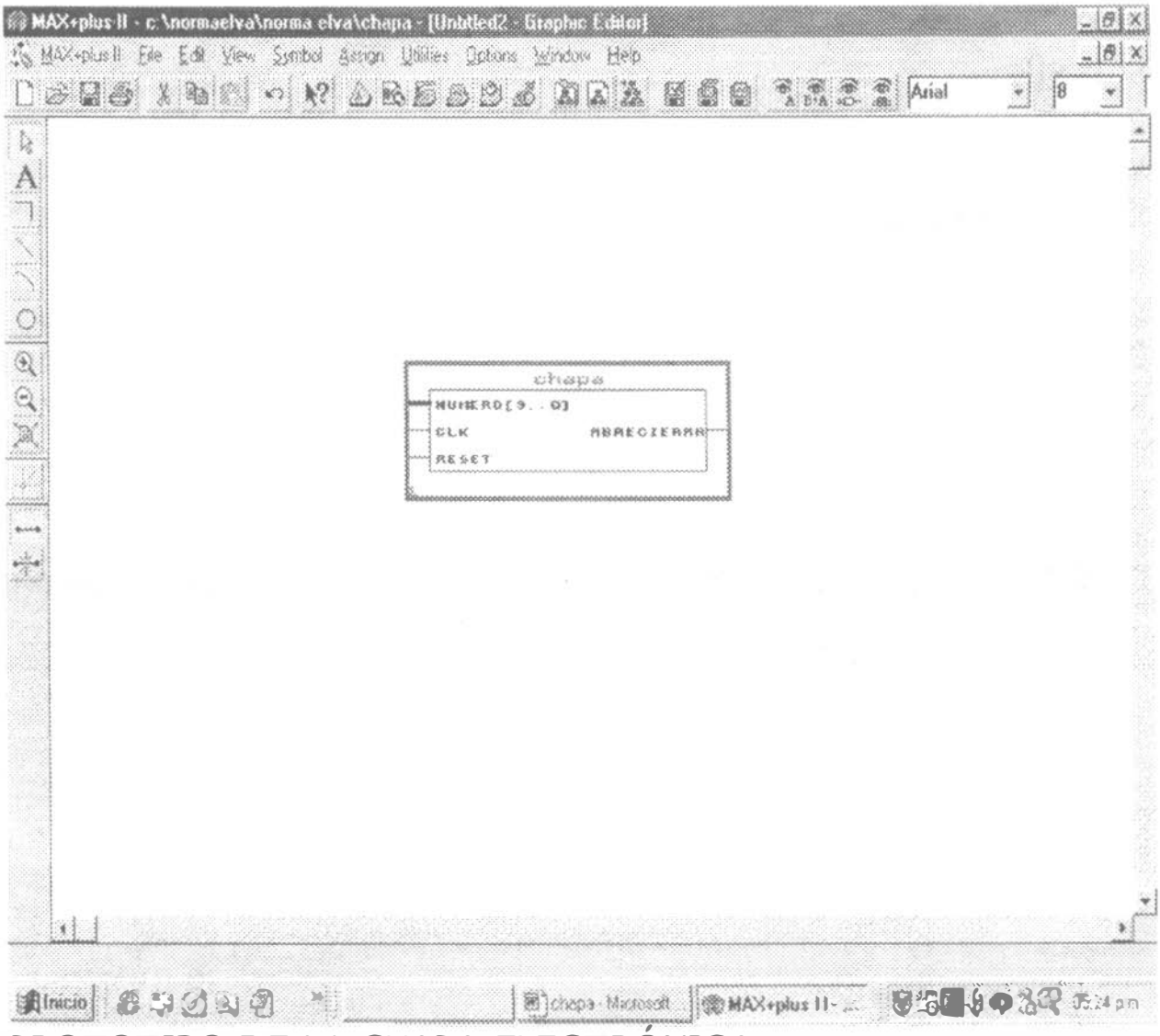
todo el programa será:

```
TITLE "CHAPA";
SUBDESIGN chapa
( numero[9..0],clk,reset:input;
abrecierra:output;)
variable
sm:machine of bits ( b3,b2,b1,b0)
with states(
s0=b"0000",
s1=b"0001",
s2=b"0010",
s3=b"0011",
s4=b"0100",
s5=b"0101");
begin
defaults
abrecierra=gnd;
end defaults;
sm.clk=clk;
sm.reset=reset;
case sm is
when s0=>
if numero1 then sm=s1;
else
sm=s0;
end if;
when s1=>
if numero2 then sm=s2;
else
sm=s0;
end if;
when s2=>
if numero7 then sm=s3;
```

```
else
sm=s0;
end if;
when s3=>
if numero9 then
abrecierra=vcc;
sm=s0;
else
sm=s0;
end if;
end case;
end;
```



SIMULACIÓN DE LA CHAPA ELECTRÒNICA



PROTOTIPO DE LA CHAPA ELECTRÓNICA.

INDICE

BIBLIOGRAFÍA

HDL Primer. J. Bhasker. Star Galaxy. Second Edition. 1999. Styles for Synthesis of Digital

Systems. David Richard Smith, Paul D. Franzon. Prentice Hall. 2001.

Logic and Computer Design Fundamentals. M. Morris Mano, Charles R. Kime. Prentice Hall. 1999.

Modern Digital Systems Design. John Y. Cheung. Ed. West. 1991.

Contemporary Logic Design.Randy H. Katz. The Benjamin /Cummings Publishing Company , Inc.1994.

indice

Esta obra se terminó de imprimir
en octubre de 2002
en el taller de imprenta del
Departamento de Publicaciones
de la Facultad de Ingeniería
Ciudad Universitaria, México, D.F.
C.P. 04510

Secretaría de Servicios Académicos

El tiraje consta de 200 ejemplares
más sobrantes de reposición

Universidad Nacional Autónoma de México

Facultad de Ingeniería