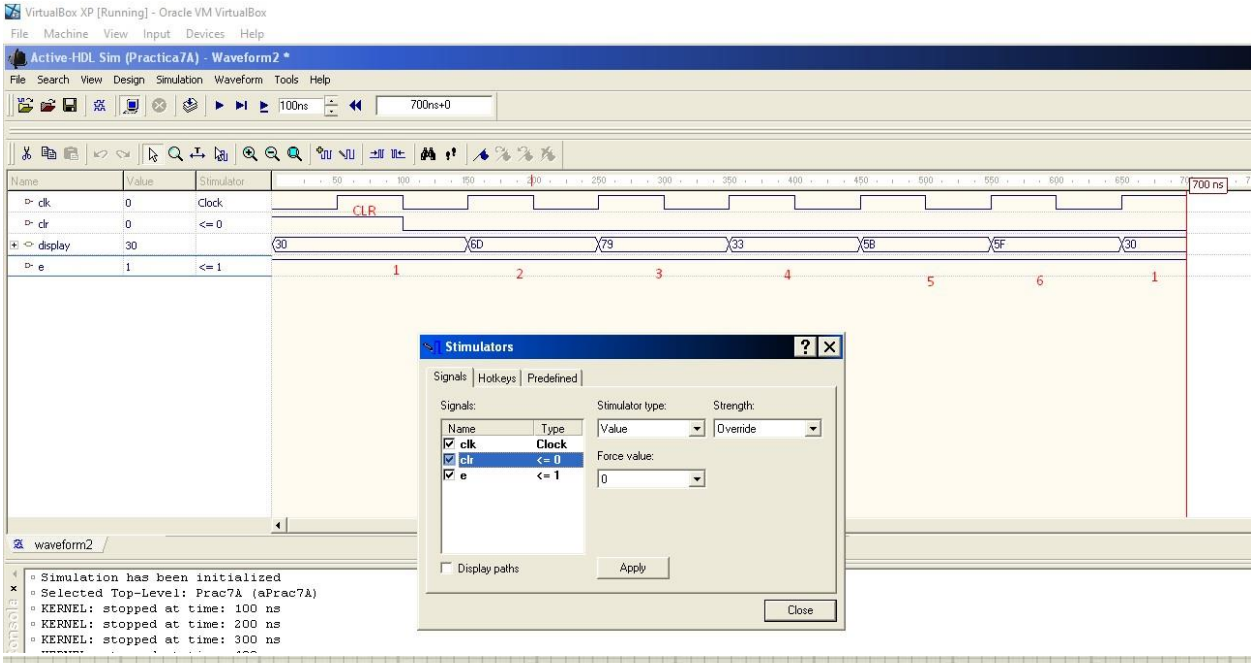
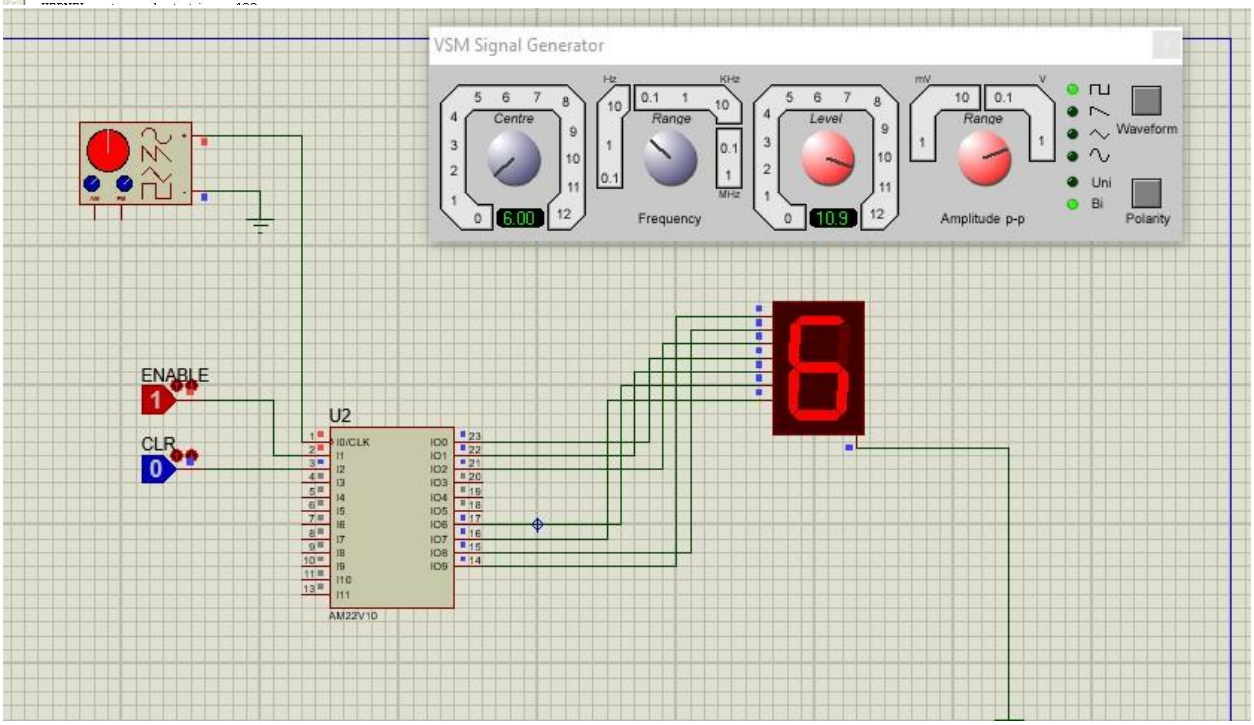
A

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

entity Prac7A is port(
	clk, clr, e : in std_logic;
	display : inout std_logic_vector(6 downto 0)
);
end Prac7A;

architecture aPrac7A of Prac7A is

begin
	process(clk, clr)
	begin
		if (clr = '1') then
			display <= "0110000";
		elsif(rising_edge(clk)) then
			if(e = '1') then
				case display is
					when "0110000" =>
						display <= "1101101";
					when "1101101" =>
						display <= "1111001";
					when "1111001" =>
						display <= "0110011";
					when "0110011" =>
						display <= "1011011";
```

```
                    when "1011011" =>
                        display <= "1011111";
                    when "1011111" =>
                        display <= "0110000";
                    when others => display <= "-------";
                end case;
            else
                display <= display;
            end if;
        end if;
    end process;

end architecture;
```


VirtualBox XP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Active-HDL Sim (Practica7A) - Waveform2 *
File Search View Design Simulation Waveform Tools Help
700ns+0
Name Value Stimulator
clk 0 Clock
clr 0 <= 0
display 30
e 1 <= 1
CLR
30 6D 79 33 5B 5F 30
700 ns
1 2 3 4 5 6 1
Stimulators
Signals Hotkeys Predefined
Signals:
Name Type
clk Clock
clr <= 0
e <= 1
Stimulator type:
Value
Strength:
Override
Force value:
0
Display paths
Apply
Close
waveform2
Simulation has been initialized
Selected Top-Level: Prac7A (aPrac7A)
KERNEL: stopped at time: 100 ns
KERNEL: stopped at time: 200 ns
KERNEL: stopped at time: 300 ns


VSM Signal Generator
Centre
6.00
Frequency
Range
Level
10.9
Amplitude p-p
Range
Waveform
Uni
Bi
Polarity
ENABLE
1
CLR
0
U2
AM22V10

Dado

Q	Q_+	E	S_2 R_2	S_1 R_1	S_0 R_0
000	001	1	0 1	1 0	0 0
001	011	1	1 1	0 1	1 0
011	010	1	1 1	1 0	0 1
010	110	1	0 1	1 0	0 1
110	111	1	1 0	1 1	0 1
111	000	1	1 0	1 1	1 1
000	000	0	0 1	1 0	0 0
001	001	0	1 1	0 1	1 0
011	011	0	1 1	1 0	0 1
010	010	0	0 1	1 0	0 1
110	110	0	1 0	1 1	0 1
111	111	0	1 0	1 1	1 1
			0 X	0 X	1 0
			0 X	1 0	X 0
			0 X	X 0	0 1
			1 0	X 0	0 X
			X 0	X 0	1 0
			0 1	0 1	0 1
			0 X	0 X	0 X
			0 X	0 X	X 0
			0 X	X 0	X 0
			0 X	X 0	0 X
			X 0	X 0	0 X
			X 0	X 0	X 0

S_2

Q_2Q_1 \ Q_0E	00	01	11	10
00	0			
01		1		
11	X	X		X
10				

$S_2 = Q_1\bar{Q}_0E$

R_2

Q_2Q_1 \ Q_0E	00	01	11	10
00	X	X	X	X
01	X		X	X
11			1	
10	X	X	X	X

$R_2 = Q_0E$

S_1

Q_2Q_1 \ Q_0E	00	01	11	10
00	0	0	1	0
01	X	X	0	X
11	X	X	0	X
10	X	X	X	X

$S_1 = \bar{Q}_2Q_0E$

R_1

Q_2Q_1 \ Q_0E	00	01	11	10
00	X	X	0	X
01	0	0	0	0
11	0	0	1	0
10	X	X	X	X

$R_1 = Q_2Q_0E$

S_0

Q_2Q_1 \ Q_0E	00	01	11	10
00	0	1	X	X
01	0	0	0	X
11	0	1	0	X
10	X	X	X	X

$S_0 = \bar{Q}_1 E + Q_2 \bar{Q}_0 E$

R_0

Q_2Q_1 \ Q_0E	00	01	11	10
00	0	0	0	0
01	X	X	1	0
11	X	0	1	0
10	X	X	X	X

$R_0 = Q_1 Q_0 E$

a

Q_2Q_1 \ Q_0E	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	1	1
10	X	X	X	X

$a = Q_2 + Q_0$

b

Q_2Q_1 \ Q_0E	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	X	X	X	X

$b = \bar{Q}_2$

c

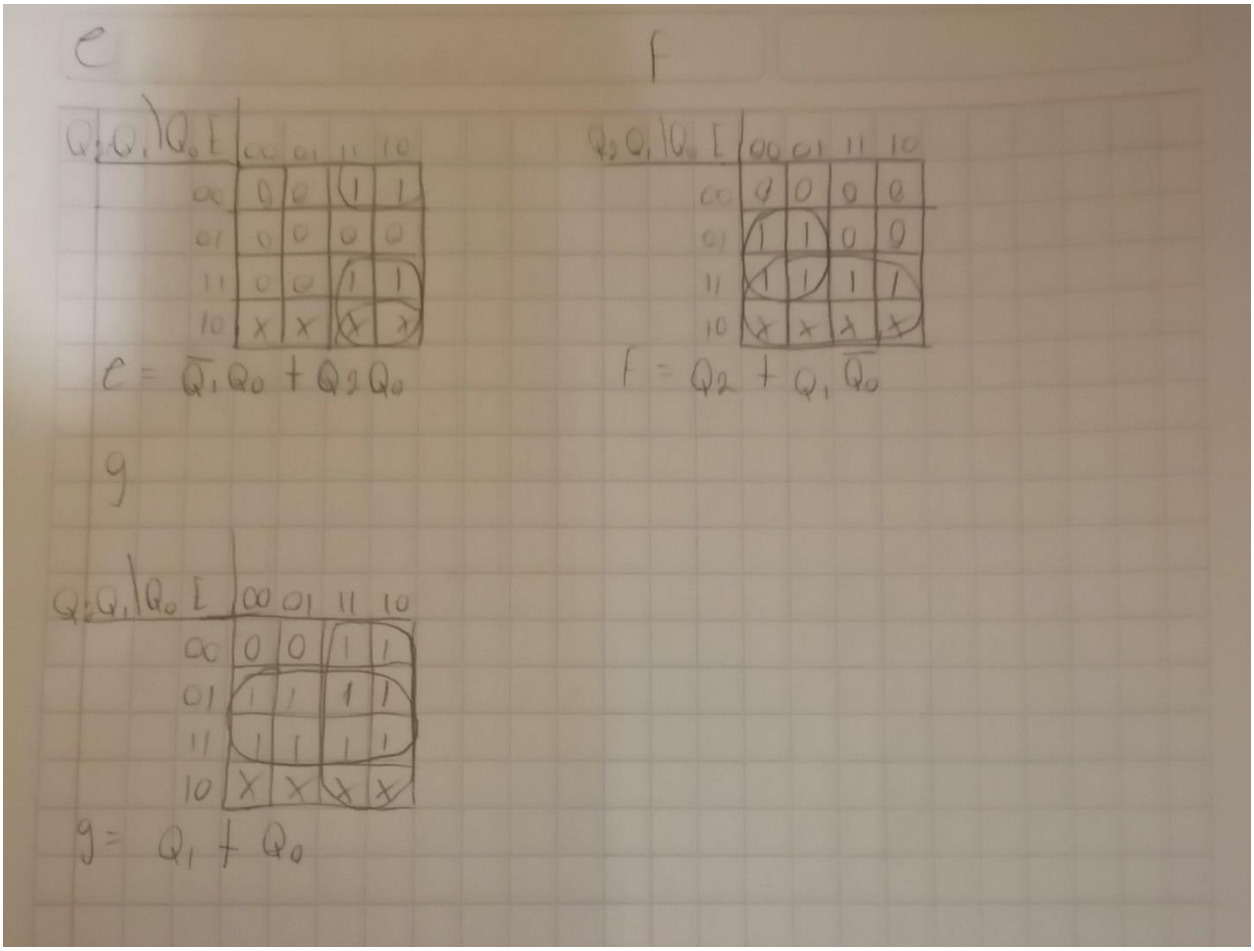
Q_2Q_1 \ Q_0E	00	01	11	10
00	1	1	0	0
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

$c = \bar{Q}_0 + Q_1$

d

Q_2Q_1 \ Q_0E	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	1	1
10	X	X	X	X

$d = Q_2 + Q_0$

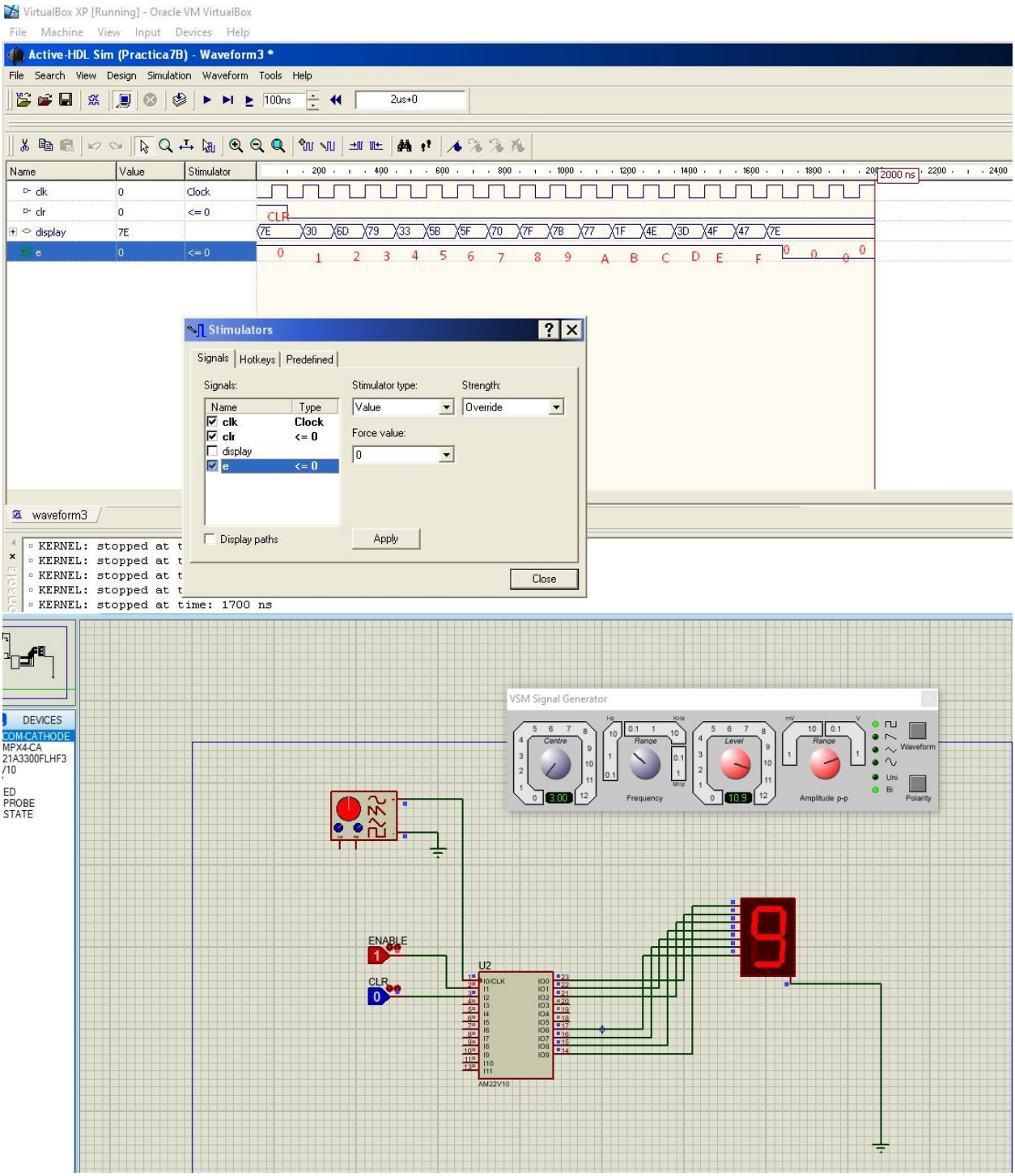


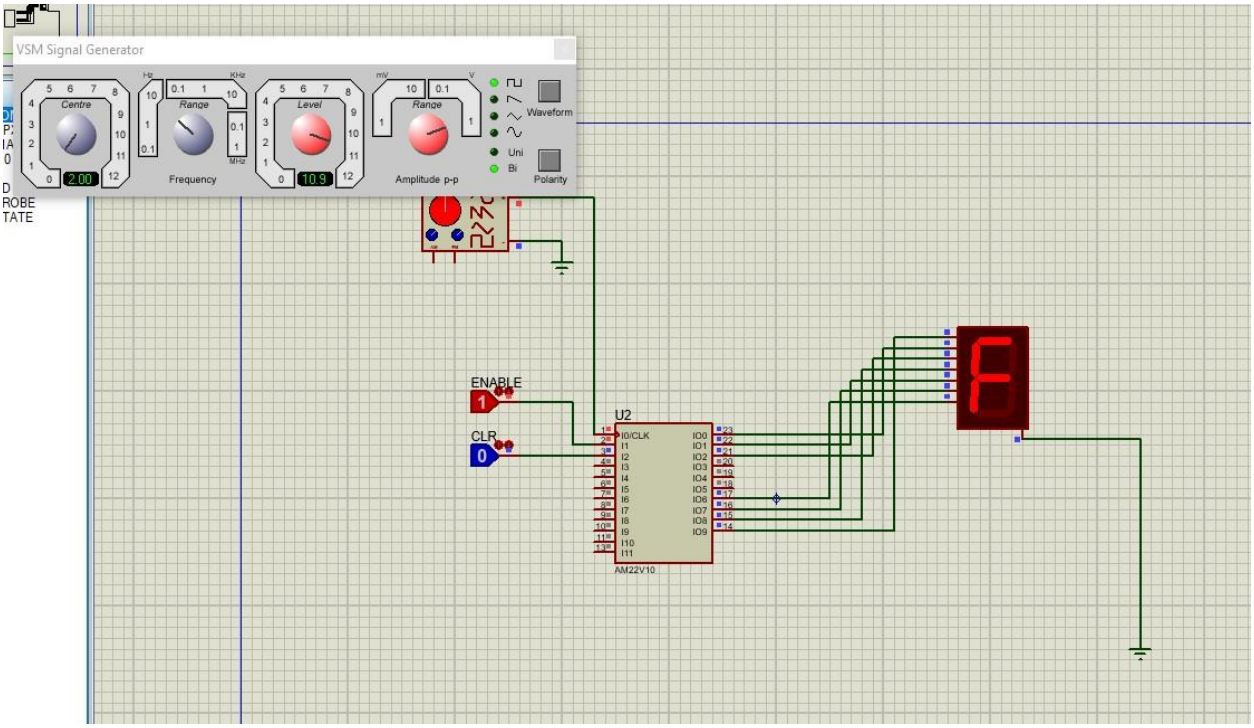
B

```
library ieee;

use ieee.std_logic_1164.all;


entity Prac7B is port(

	clk, clr, e : in std_logic;

	display : inout std_logic_vector(6 downto 0)

);

end Prac7B;
```

```
architecture aPrac7B of Prac7B is


begin
	process(clk, clr)
	begin
		if (clr = '1') then
			display <= "1111110"; --0
		elsif(rising_edge(clk)) then
			if(e = '1') then
				case display is
					when "1111110" => --1
						display <= "0110000";
					when "0110000" => --2
						display <= "1101101";
					when "1101101" => --3
						display <= "1111001";
					when "1111001" => --4
						display <= "0110011";
					when "0110011" => --5
						display <= "1011011";
					when "1011011" => --6
						display <= "1011111";
					when "1011111" => --7
						display <= "1110000";
					when "1110000" => --8
						display <= "1111111";
					when "1111111" => --9
						display <= "1111011";
					when "1111011" => --A
```

```
                    display <= "1110111";
                when "1110111" => --B
                    display <= "0011111";
                when "0011111" => --C
                    display <= "1001110";
                when "1001110" => --D
                    display <= "0111101";
                when "0111101" => --E
                    display <= "1001111";
                when "1001111" => --F
                    display <= "1000111";
                when "1000111" => --0
                    display <= "1111110";
                when others => display <= "-------";
            end case;
        else
            display <= display;
        end if;
    end if;
end process;

end architecture;
```


VirtualBox XP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Active-HDL Sim (Practica7B) - Waveform3 *
File Search View Design Simulation Waveform Tools Help
Name Value Stimulator
clk 0 Clock
clr 0 <= 0
display 7E
e 0 <= 0
2000 ns
Stimulators
Signals Hotkeys Predefined
Signals:
Name Type
clk Clock
clr <= 0
display
e <= 0
Stimulator type:
Value
Strength:
Override
Force value:
0
Display paths
Apply
Close
waveform3
KERNEL: stopped at time: 1700 ns
DEVICES
COM-CATHODE
MPX4-CA
21A3300FLHF3
/10
ED
PROBE
STATE
VSM Signal Generator
Centre
Range
Frequency
Level
Range
Amplitude p-p
Waveform
Uni
Bi
Polarity
ENABLE
CLR
U2
AM22V10

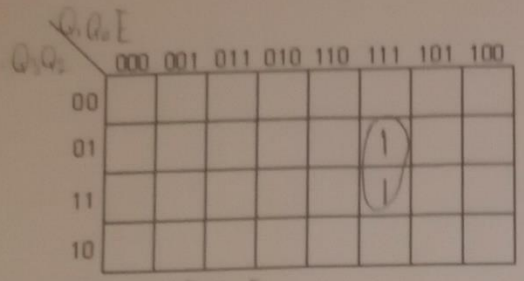

VSM Signal Generator
Centre
Range
Frequency
Level
Range
Amplitude p-p
Waveform
Uni
Bi
Polarity
2.00
10.9
ENABLE
CLR
U2
AM22V10

Contador hexadecimal

Q	Q_1	I	Salida		J_3	J_2	J_1	J_0
0 0 0 0	0 0 0 1	1	1 1 1 1 1 1 0	0	0	0	0	1
0 0 0 1	0 0 1 0	1	0 1 1 0 0 0 0	1	0	0	1	1
0 0 1 0	0 0 1 1	1	1 1 0 1 1 0 1	2	0	0	0	1
0 0 1 1	0 1 0 0	1	1 1 1 1 0 0 1	3	0	1	1	1
0 1 0 0	0 1 0 1	1	0 1 1 0 0 1 1	4	0	0	0	1
0 1 0 1	0 1 1 0	1	1 0 1 1 0 1 1	5	0	0	1	1
0 1 1 0	0 1 1 1	1	1 0 1 1 1 1 1	6	0	0	0	1
0 1 1 1	1 0 0 0	1	1 1 1 0 0 0 0	7	1	1	1	1
1 0 0 0	1 0 0 1	1	1 1 1 1 1 1 1	8	0	0	0	1
1 0 0 1	1 0 1 0	1	1 1 1 1 0 1 1	9	0	0	1	1
1 0 1 0	1 0 1 1	1	1 1 1 0 1 1 1	A	0	0	0	1
1 0 1 1	1 1 0 0	1	0 0 1 1 1 1 1	b	0	1	1	1
1 1 0 0	1 1 0 1	1	1 0 0 1 1 1 0	C	0	0	0	1
1 1 0 1	1 1 1 0	1	0 1 1 1 1 0 1	d	0	0	1	1
1 1 1 0	1 1 1 1	1	1 0 0 1 1 1 1	E	0	0	0	1
1 1 1 1	0 0 0 0	1	1 0 0 0 1 1 1	f	1	1	1	1
0 0 0 0	0 0 0 0	0	1 1 1 1 1 1 0	0	0	0	0	0
0 0 0 1	0 0 0 1	0	0 1 1 0 0 0 0	1	0	0	0	0
0 0 1 0	0 0 1 0	0	1 1 0 1 1 0 1	2	0	0	0	0
0 0 1 1	0 0 1 1	0	1 1 1 1 0 0 1	3	0	0	0	0
0 1 0 0	0 1 0 0	0	0 1 1 0 0 1 1	4	0	0	0	0
0 1 0 1	0 1 0 1	0	1 0 1 1 0 1 1	5	0	0	0	0
0 1 1 0	0 1 1 0	0	1 0 1 1 1 1 0	6	0	0	0	0
0 1 1 1	0 1 1 1	0	1 1 1 0 0 0 0	7	0	0	0	0
1 0 0 0	1 0 0 0	0	1 1 1 1 1 1 1	8	0	0	0	0
1 0 0 1	1 0 0 1	0	1 1 1 1 0 1 1	9	0	0	0	0
1 0 1 0	1 0 1 0	0	1 1 1 0 1 1 1	A	0	0	0	0
1 0 1 1	1 0 1 1	0	0 0 1 1 1 1 1	b	0	0	0	0
1 1 0 0	1 1 0 0	0	1 0 0 1 1 1 0	C	0	0	0	0
1 1 0 1	1 1 0 1	0	0 1 1 1 1 0 1	d	0	0	0	0
1 1 1 0	1 1 1 0	0	1 0 0 1 1 1 1	E	0	0	0	0
1 1 1 1	1 1 1 1	0	1 0 0 0 1 1 1	F	0	0	0	0

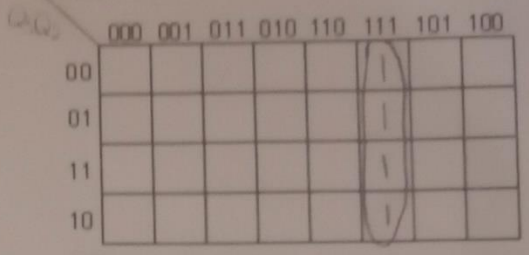
Contador hexadecimal

T_3



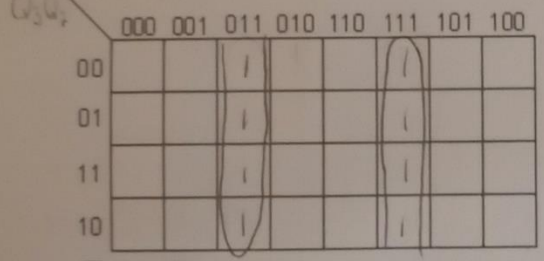
$T_3 = Q_3Q_1Q_0E$

T_2



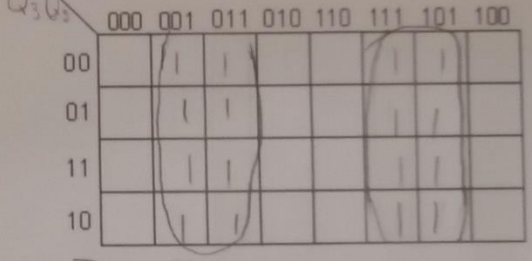
$T_2 = Q_1Q_0E$

T_1

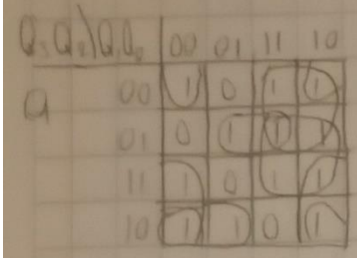


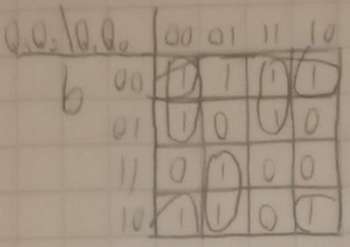
$T_1 = Q_0E$

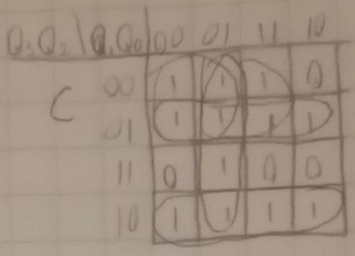
T_0

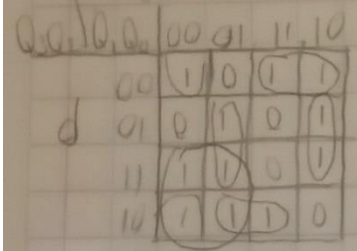


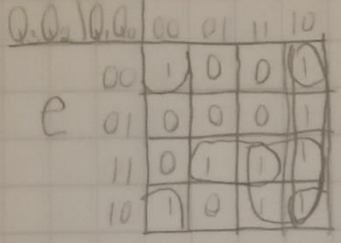
$T_0 = E$

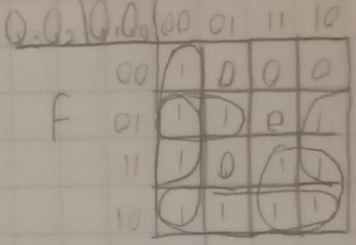


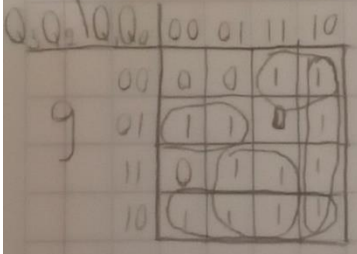












$a = \bar{Q}_2\bar{Q}_0 + \bar{Q}_3Q_1 + Q_2Q_1 + Q_3\bar{Q}_1 + \bar{Q}_3Q_2Q_0 + Q_3\bar{Q}_2\bar{Q}_0$

$b = \bar{Q}_3\bar{Q}_2 + \bar{Q}_2\bar{Q}_0 + \bar{Q}_3\bar{Q}_1\bar{Q}_0 + \bar{Q}_3Q_1Q_0 + Q_3\bar{Q}_1Q_0$

$c = \bar{Q}_3\bar{Q}_1 + \bar{Q}_3Q_0 + \bar{Q}_1Q_0 + \bar{Q}_3Q_2 + Q_3\bar{Q}_2$

$d = Q_3\bar{Q}_1 + \bar{Q}_3\bar{Q}_2\bar{Q}_1 + \bar{Q}_2Q_1Q_0 + Q_2\bar{Q}_1Q_0 + Q_2Q_1\bar{Q}_0$

$e = \bar{Q}_2\bar{Q}_0 + Q_1\bar{Q}_0 + Q_3Q_1 + Q_3Q_2Q_0$

$f = \bar{Q}_1\bar{Q}_0 + Q_2\bar{Q}_0 + Q_3\bar{Q}_2 + Q_3Q_1 + \bar{Q}_3Q_2\bar{Q}_1$

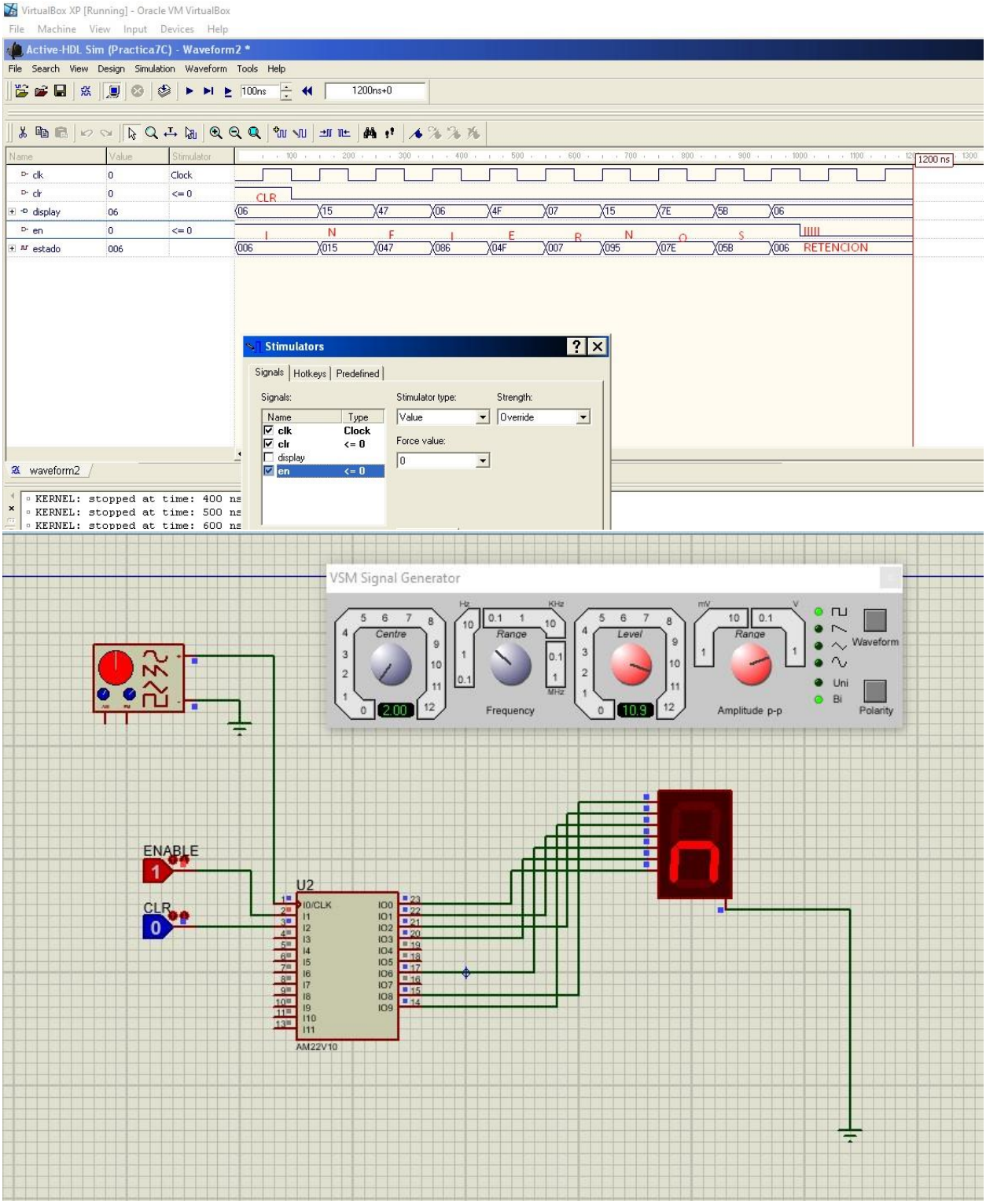
$g = \bar{Q}_2Q_1 + Q_1\bar{Q}_0 + Q_3\bar{Q}_2 + Q_3Q_0 + \bar{Q}_3Q_2\bar{Q}_1$

C

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

entity Prac7B is port(
	clk, clr, en : in std_logic;
	display : out std_logic_vector(6 downto 0)
);
end Prac7B;

architecture aPrac7B of Prac7B is
constant l0 : std_logic_vector(1 downto 0) := "00";
constant l1 : std_logic_vector(1 downto 0) := "01";
constant l2 : std_logic_vector(1 downto 0) := "10";

constant i : std_logic_vector(6 downto 0) := "0000110";
constant n : std_logic_vector(6 downto 0) := "0010101";
constant f : std_logic_vector(6 downto 0) := "1000111";
constant e : std_logic_vector(6 downto 0) := "1001111";
constant r : std_logic_vector(6 downto 0) := "0000111";
constant o : std_logic_vector(6 downto 0) := "1111110";
constant s : std_logic_vector(6 downto 0) := "1011011";

signal estado : std_logic_vector(8 downto 0);

begin
	process(clk, clr)
	begin
```

```
if (clr = '1') then
        estado <= l0&i;
elsif(rising_edge(clk)) then
        if(en = '1') then
                case estado is
                        when "000000110" =>
                                estado <= "000010101";
                        when "000010101" =>
                                estado <= "001000111";
                        when "001000111" =>
                                estado <= "010000110";
                        when "010000110" =>
                                estado <= "001001111";
                        when "001001111" =>
                                estado <= "000000111";
                        when "000000111" =>
                                estado <= "010010101";
                        when "010010101" =>
                                estado <= "001111110";
                        when "001111110" =>
                                estado <= "001011011";
                        when "001011011" =>
                                estado <= "000000110";
                        when others => estado <= "---------";
                end case;
         else
                estado <= estado;
         end if;
end if;
```

```
        end process;


        display <= estado(6 downto 0); -- para solo tomar los bits que pertenecen a la letra e ignorar la etiqueta de repeticion

end architecture;
```

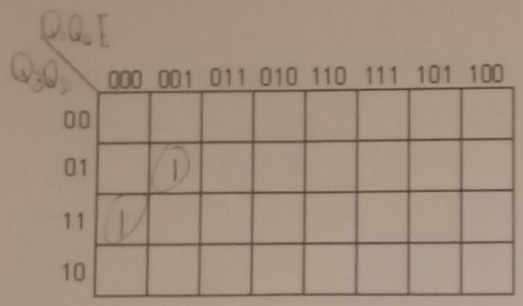

VirtualBox XP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Active-HDL Sim (Practica7C) - Waveform2 *
File Search View Design Simulation Waveform Tools Help
Name Value Stimulator
clk 0 Clock
clr 0 <= 0
display 06
en 0 <= 0
estado 006
CLR
RETENCION
Stimulators
Signals Hotkeys Predefined
Signals:
Name Type
clk Clock
clr <= 0
display
en <= 0
Stimulator type:
Value
Strength:
Override
Force value:
0
waveform2
KERNEL: stopped at time: 400 ns
KERNEL: stopped at time: 500 ns
KERNEL: stopped at time: 600 ns
VSM Signal Generator
Centre
Range
Frequency
Level
Range
Amplitude p-p
Waveform
Uni
Bi
Polarity
2.00
10.9
ENABLE
CLR
U2
AM22V10

Contador de mensaje

Q	Q1	E	Salida								D_3	D_2	D_1	D_0
0000	0001	1	0	0	0	0	1	1	0	I	0	0	0	1
0001	0011	1	0	0	1	0	1	0	1	n	0	0	1	1
0011	0010	1	1	0	0	0	1	1	1	F	0	0	1	0
0010	0110	1	0	0	0	0	1	1	0	I	0	1	1	0
0110	0111	1	1	0	0	1	1	1	1	E	0	1	1	1
0111	0101	1	0	0	0	0	1	1	1	t	0	1	0	1
0101	0100	1	0	0	1	0	1	0	1	n	0	1	0	0
0100	1100	1	1	1	1	1	1	1	0	O	1	1	0	0
1100	0000	1	1	0	1	1	0	1	1	S	0	0	0	0
0000	0000	0	0	0	0	0	1	1	0	I	0	0	0	0
0001	0001	0	0	0	1	0	1	0	1	n	0	0	0	1
0011	0011	0	1	0	0	0	1	1	1	T	0	0	1	1
0010	0010	0	0	0	0	0	1	1	0	I	0	0	1	0
0110	0110	0	1	0	0	1	1	1	1	E	0	1	1	0
0111	0111	0	0	0	0	0	1	1	1	t	0	1	1	1
0101	0101	0	0	0	1	0	1	0	1	n	0	1	0	1
0100	0100	0	1	1	1	1	1	1	0	O	0	1	0	0
1100	1100	0	1	0	1	1	0	1	1	S	1	1	0	0

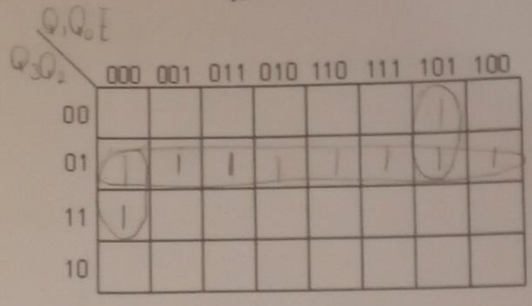
Contador mensaje

D_3



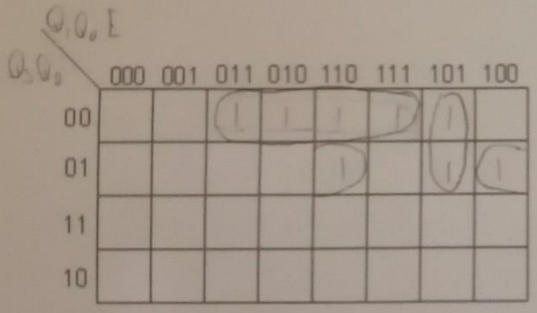
$D_3 = \bar{Q}_3\bar{Q}_2\bar{Q}_1\bar{Q}_0E + Q_3Q_2\bar{Q}_1\bar{Q}_0\bar{E}$

D_2



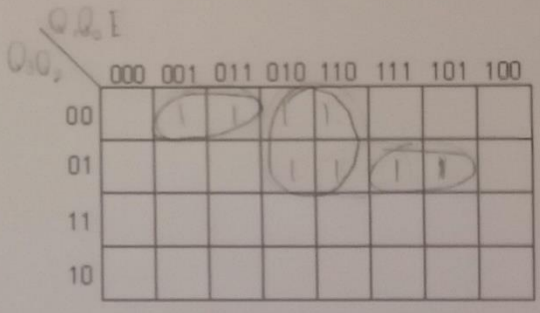
$D_2 = \bar{Q}_3Q_2 + \bar{Q}_3Q_1\bar{Q}_0E + Q_2\bar{Q}_1\bar{Q}_0\bar{E}$

D_1

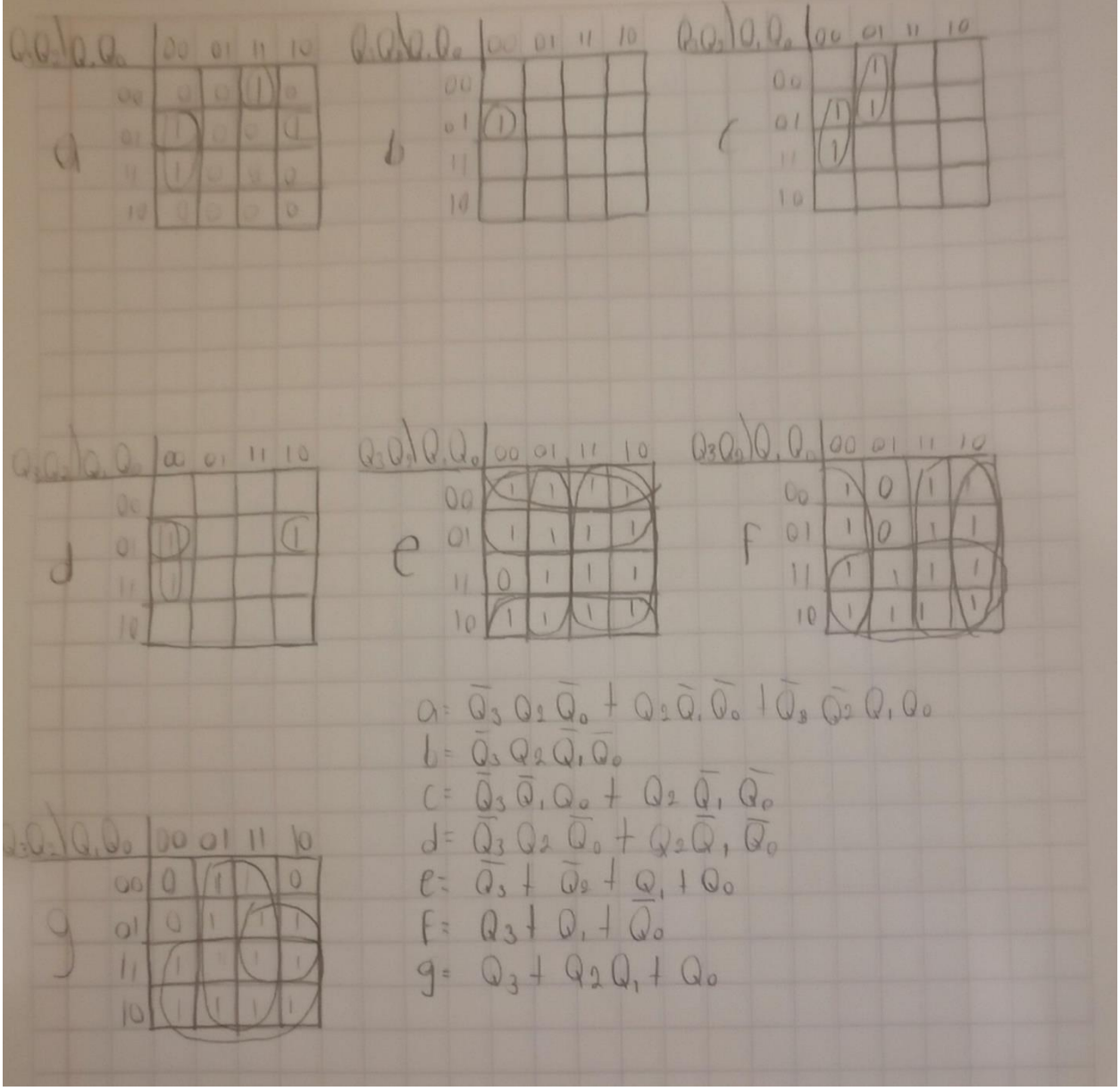


$D_1 = \bar{Q}_3\bar{Q}_2Q_0 + \bar{Q}_3Q_1\bar{Q}_0E + \bar{Q}_3Q_2Q_1\bar{E}$

D_0



$D_0 = \bar{Q}_3Q_0\bar{E} + \bar{Q}_3\bar{Q}_2\bar{Q}_1E + \bar{Q}_3Q_2Q_1E$



$a = \bar{Q}_3 Q_2 \bar{Q}_0 + Q_2 \bar{Q}_1 \bar{Q}_0 + \bar{Q}_3 \bar{Q}_2 Q_1 Q_0$

$b = \bar{Q}_3 Q_2 \bar{Q}_1 \bar{Q}_0$

$c = \bar{Q}_3 \bar{Q}_1 Q_0 + Q_2 \bar{Q}_1 \bar{Q}_0$

$d = \bar{Q}_3 Q_2 \bar{Q}_0 + Q_2 \bar{Q}_1 \bar{Q}_0$

$e = \bar{Q}_3 + \bar{Q}_2 + Q_1 + Q_0$

$F = Q_3 + Q_1 + \bar{Q}_0$

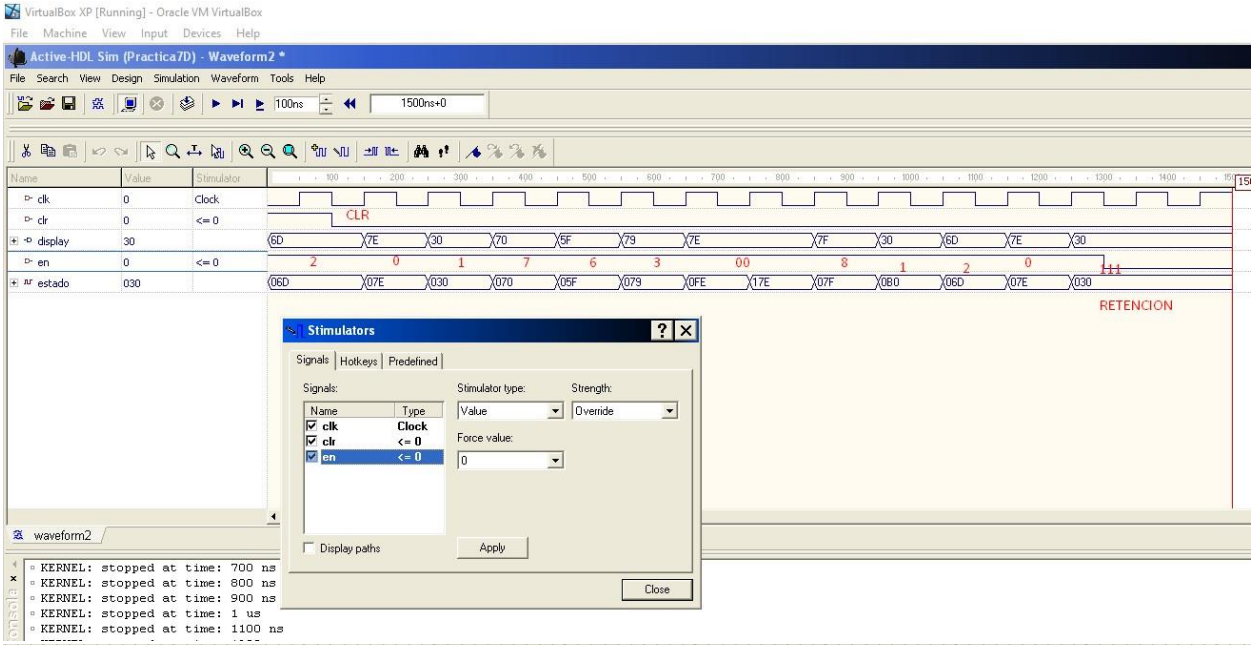
$g = Q_3 + Q_2 Q_1 + Q_0$

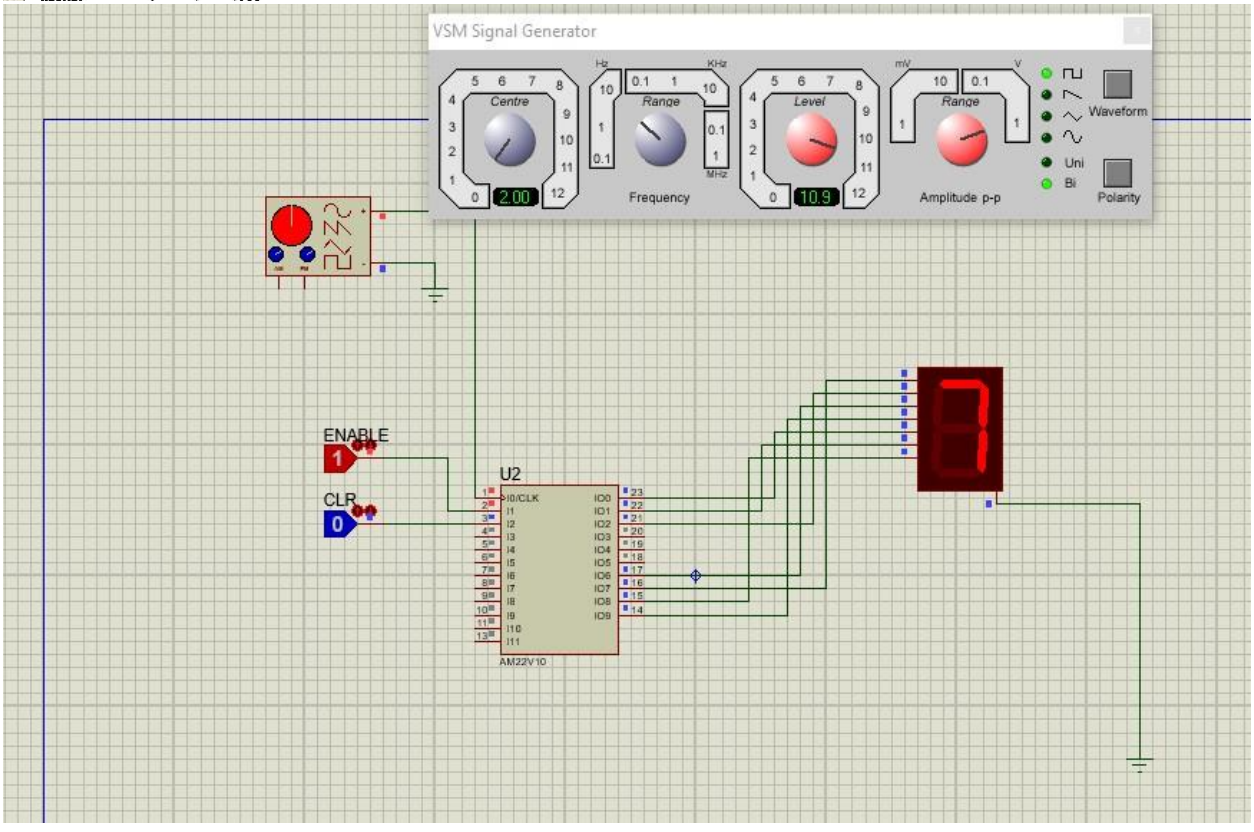
D

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

entity Prac7B is port(
	clk, clr, en : in std_logic;
	display : out std_logic_vector(6 downto 0)
);
end Prac7B;

architecture aPrac7B of Prac7B is
constant l0 : std_logic_vector(1 downto 0) := "00";
constant l1 : std_logic_vector(1 downto 0) := "01";
constant l2 : std_logic_vector(1 downto 0) := "10";

constant n2 : std_logic_vector(6 downto 0) := "1101101";
constant n0 : std_logic_vector(6 downto 0) := "1111110";
constant n1 : std_logic_vector(6 downto 0) := "0110000";
constant n7 : std_logic_vector(6 downto 0) := "1110000";
constant n6 : std_logic_vector(6 downto 0) := "1011111";
constant n3 : std_logic_vector(6 downto 0) := "1111001";
constant n8 : std_logic_vector(6 downto 0) := "1111111";

signal estado : std_logic_vector(8 downto 0);

begin
	process(clk, clr)
	begin
```

```
if (clr = '1') then
        estado <= l0&"1101101";
elsif(rising_edge(clk)) then
        if(en = '1') then
                case estado is
                        when "001101101" =>
                                estado <= "001111110";
                        when "001111110" =>
                                estado <= "000110000";
                        when "000110000" =>
                                estado <= "001110000";
                        when "001110000" =>
                                estado <= "001011111";
                        when "001011111" =>
                                estado <= "001111001";
                        when "001111001" =>
                                estado <= "011111110";
                        when "011111110" =>
                                estado <= "101111110";
                        when "101111110" =>
                                estado <= "001111111";
                        when "001111111" =>
                                estado <= "010110000";
                        when "010110000" =>
                                estado <= "001101101";
                        when others => estado <= "---------";
                end case;
         else
                estado <= estado;
```

```
                    end if;
                end if;
        end process;

        display <= estado(6 downto 0); -- para solo tomar los bits que pertenecen a la letra e ignorar la etiqueta de repeticion
end architecture;
```


VirtualBox XP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Active-HDL Sim (Practica7D) - Waveform2 *
File Search View Design Simulation Waveform Tools Help
clk
clr
display
en
estado
CLR
RETENCION
Stimulators
Signals
Hotkeys
Predefined
Signals:
Name
Type
clk
Clock
clr
<= 0
en
<= 0
Stimulator type:
Value
Strength:
Override
Force value:
0
Display paths
Apply
Close
waveform2
KERNEL: stopped at time: 700 ns
KERNEL: stopped at time: 800 ns
KERNEL: stopped at time: 900 ns
KERNEL: stopped at time: 1 us
KERNEL: stopped at time: 1100 ns

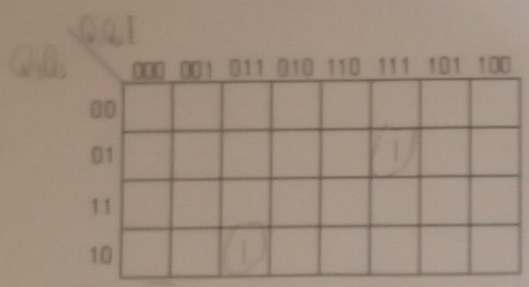

VSM Signal Generator
Centre
2.00
Frequency
Level
10.9
Amplitude p-p
Range
Waveform
Uni
Bi
Polarity
ENABLE
1
CLR
0
U2
AM22V10

Contador de boleta

Q	Q+	E	Salida								T_3	T_2	T_1	T_0
0 0 0 0	0 0 0 1	1	1	1	0	1	1	0	1	2	0	0	0	1
0 0 0 1	0 0 1 0	1	1	1	1	1	1	1	0	0	0	0	1	1
0 0 1 0	0 0 1 1	1	0	1	1	0	0	0	0	1	0	0	0	1
0 0 1 1	0 1 0 0	1	1	1	1	0	0	0	0	7	0	1	1	1
0 1 0 0	0 1 0 1	1	1	0	1	1	1	1	1	6	0	0	0	1
0 1 0 1	0 1 1 0	1	1	1	1	1	0	0	1	3	0	0	1	1
0 1 1 0	0 1 1 1	1	1	1	1	1	1	1	0	0	0	0	0	1
0 1 1 1	1 0 0 0	1	1	1	1	1	1	1	0	0	1	1	1	1
1 0 0 0	1 0 0 1	1	1	1	1	1	1	1	1	8	0	0	0	1
1 0 0 1	0 0 0 0	1	0	1	1	0	0	0	0	1	1	0	0	1
0 0 0 0	0 0 0 0	0	1	1	0	1	1	0	1	2	0	0	0	0
0 0 0 1	0 0 0 1	0	1	1	1	1	1	1	0	0	0	0	0	0
0 0 1 0	0 0 1 0	0	0	1	1	0	0	0	0	1	0	0	0	0
0 0 1 1	0 0 1 1	0	1	1	1	0	0	0	0	7	0	0	0	0
0 1 0 0	0 1 0 0	0	1	0	1	1	1	1	1	6	0	0	0	0
0 1 0 1	0 1 0 1	0	1	1	1	1	0	0	1	3	0	0	0	0
0 1 1 0	0 1 1 0	0	1	1	1	1	1	1	0	0	0	0	0	0
0 1 1 1	0 1 1 1	0	1	1	1	1	1	1	0	0	0	0	0	0
1 0 0 0	1 0 0 0	0	1	1	1	1	1	1	1	8	0	0	0	0
1 0 0 1	1 0 0 1	0	1	1	1	0	0	0	0	1	0	0	0	0

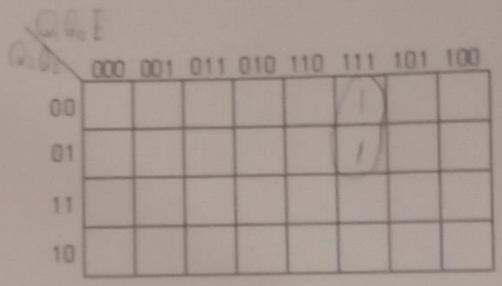
Contador boleta

T_3



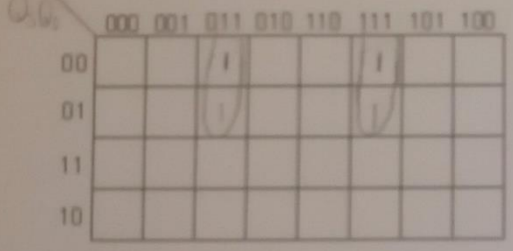
$T_3 = \bar{Q}_3 Q_2 Q_1 Q_0 I + Q_3 \bar{Q}_2 \bar{Q}_1 Q_0 I$

T_2



$T_2 = \bar{Q}_3 Q_1 Q_0 I$

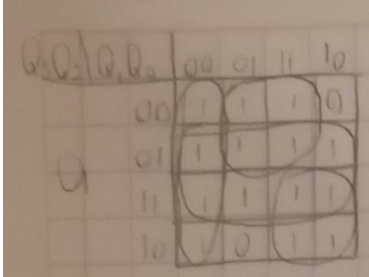
T_1

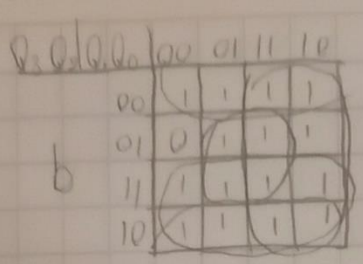


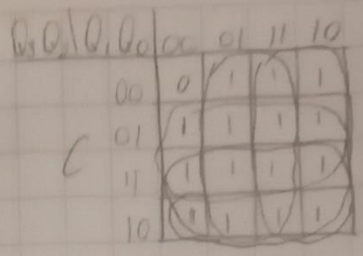
$T_1 = \bar{Q}_3 Q_0 I$

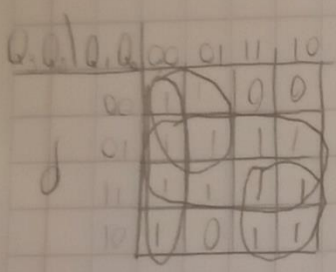
T_0

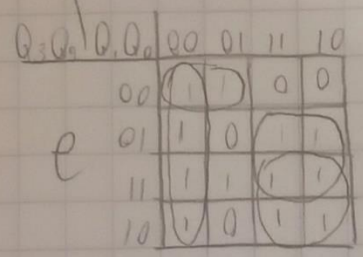
$T_0 = I$

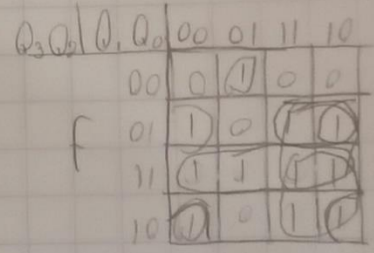


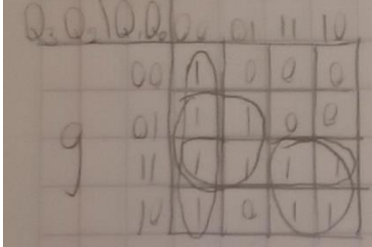












$a = Q_2 + \bar{Q}_3\bar{Q}_1 + Q_1Q_0 + Q_3\bar{Q}_0$

$b = \bar{Q}_2 + Q_3 + Q_1 + Q_0$

$c = Q_3 + Q_2 + Q_1 + Q_0$

$d = Q_2 + \bar{Q}_3\bar{Q}_1 + \bar{Q}_1\bar{Q}_0 + Q_3Q_1$

$e = \bar{Q}_1\bar{Q}_0 + Q_2Q_1 + Q_3Q_1 + Q_3Q_2 + \bar{Q}_3\bar{Q}_2\bar{Q}_1$

$f = Q_2\bar{Q}_0 + Q_2Q_1 + Q_3\bar{Q}_0 + Q_3Q_1 + Q_3Q_2 + \bar{Q}_3\bar{Q}_2\bar{Q}_1Q_0$

$g = \bar{Q}_1\bar{Q}_0 + Q_2\bar{Q}_1 + Q_3Q_1$

CUESTIONARIO

1. ¿Cuántos dispositivos PLD 22V10 son necesarios para el desarrollo de esta práctica?

1 para cada contador, 4 en total

2. ¿Cuántos dispositivos de la serie 74xx (TTL) ó 40xx (CMOS) hubieras necesitado para el desarrollo de esta práctica?

Mas de 100

3. ¿Cuántos pines de entrada/salida del PLD 22V10 se usan en los diseños?

43 entre los 4 autómatas, la mayoría utiliza las macroceldas de input/output

4. ¿Cuántos términos producto ocupan las ecuaciones para cada señal de salida y que porcentaje se usa en total del PLD 22V10 en cada aplicación?

A – 17%

B – 28%

C - 18%

D – 29%

5. ¿Es posible implementar los diseños usando cualquier tipo de codificación en el PLD22V10?

Sí, hay muchas maneras de codificar estos autómatas, pero se deben de tener en cuenta las limitaciones físicas de al GAL

6. ¿Cuáles son las señales que funcionan de manera síncrona y cuales de manera asíncrona?

La señal asíncrona es el “CLR”, todas las demás son asíncronas

7. ¿Qué puedes concluir de esta práctica?

Mediante el uso de contadores se pueden crear autómatas que modelan una cantidad increíble de cosas, esta práctica fue pesada en el sentido de que el análisis a mano puede complicarse exponencialmente cuando se aumenta el número de bits necesarios para representar algo mediante un autómata.