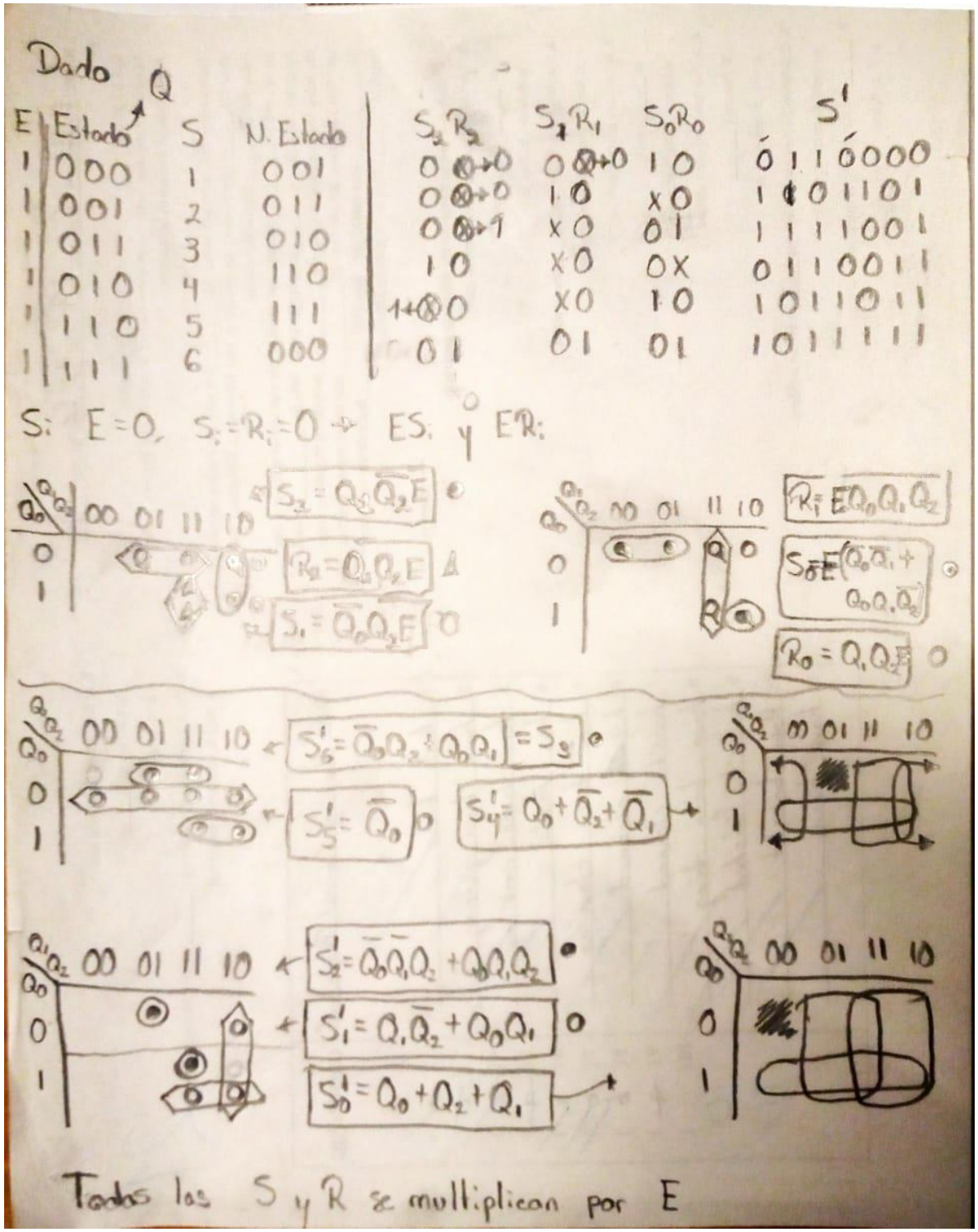
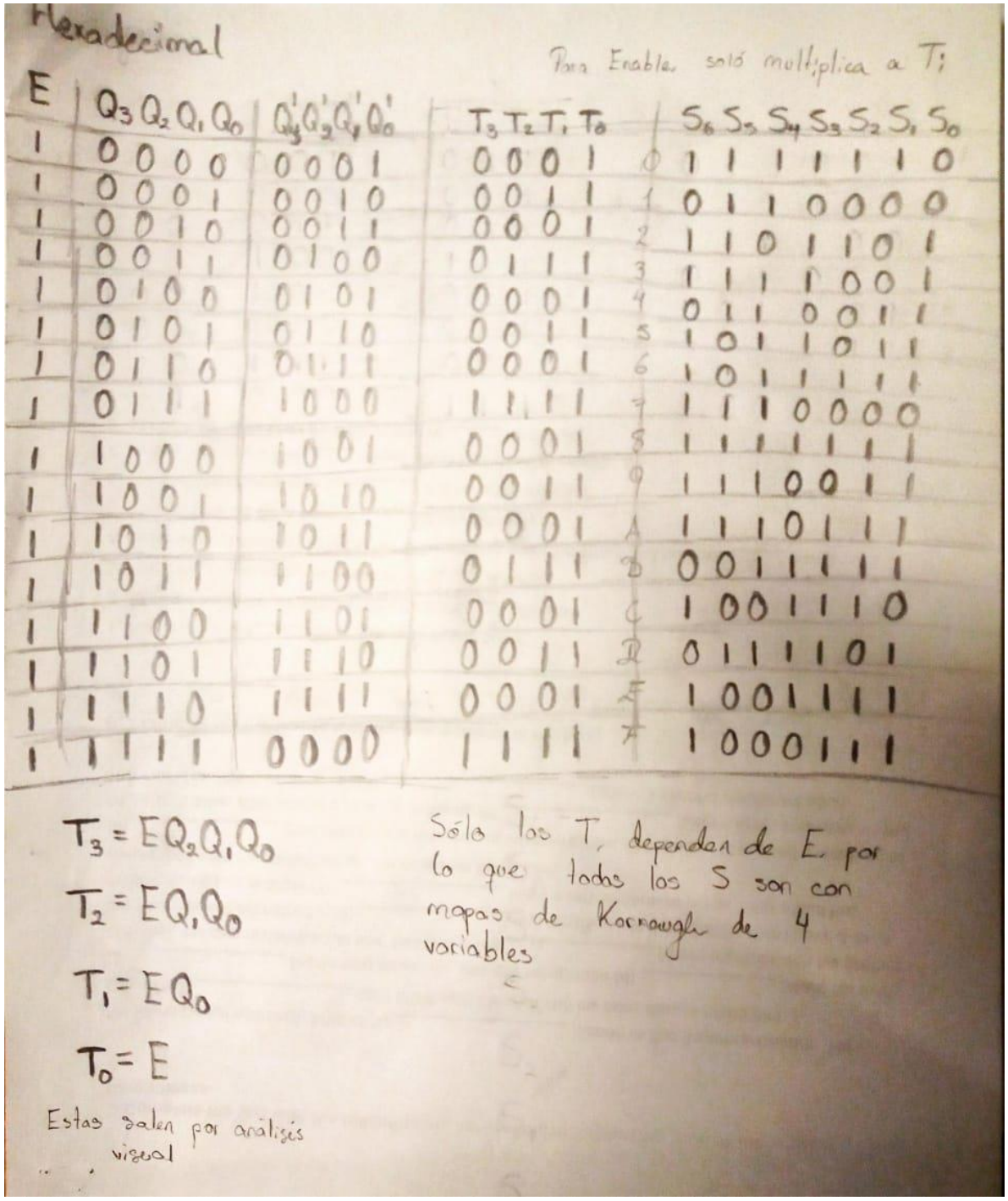
Martínez Coronel Brayan Yosafat

ANÁLISIS

DADO



HEXADECIMAL



Hexadecimal

Para Enable solo multiplica a T_i

E	Q_3 Q_2 Q_1 Q_0	Q_3' Q_2' Q_1' Q_0'	T_3 T_2 T_1 T_0		S_6 S_5 S_4 S_3 S_2 S_1 S_0
1	0 0 0 0	0 0 0 1	0 0 0 1	0	1 1 1 1 1 1 0
1	0 0 0 1	0 0 1 0	0 0 1 1	1	0 1 1 0 0 0 0
1	0 0 1 0	0 0 1 1	0 0 0 1	2	1 1 0 1 1 0 1
1	0 0 1 1	0 1 0 0	0 1 1 1	3	1 1 1 1 0 0 1
1	0 1 0 0	0 1 0 1	0 0 0 1	4	0 1 1 0 0 1 1
1	0 1 0 1	0 1 1 0	0 0 1 1	5	1 0 1 1 0 1 1
1	0 1 1 0	0 1 1 1	0 0 0 1	6	1 0 1 1 1 1 1
1	0 1 1 1	1 0 0 0	1 1 1 1	7	1 1 1 0 0 0 0
1	1 0 0 0	1 0 0 1	0 0 0 1	8	1 1 1 1 1 1 1
1	1 0 0 1	1 0 1 0	0 0 1 1	9	1 1 1 0 0 1 1
1	1 0 1 0	1 0 1 1	0 0 0 1	A	1 1 1 0 1 1 1
1	1 0 1 1	1 1 0 0	0 1 1 1	B	0 0 1 1 1 1 1
1	1 1 0 0	1 1 0 1	0 0 0 1	C	1 0 0 1 1 1 0
1	1 1 0 1	1 1 1 0	0 0 1 1	D	0 1 1 1 1 0 1
1	1 1 1 0	1 1 1 1	0 0 0 1	E	1 0 0 1 1 1 1
1	1 1 1 1	0 0 0 0	1 1 1 1	F	1 0 0 0 1 1 1

$T_3 = EQ_2Q_1Q_0$

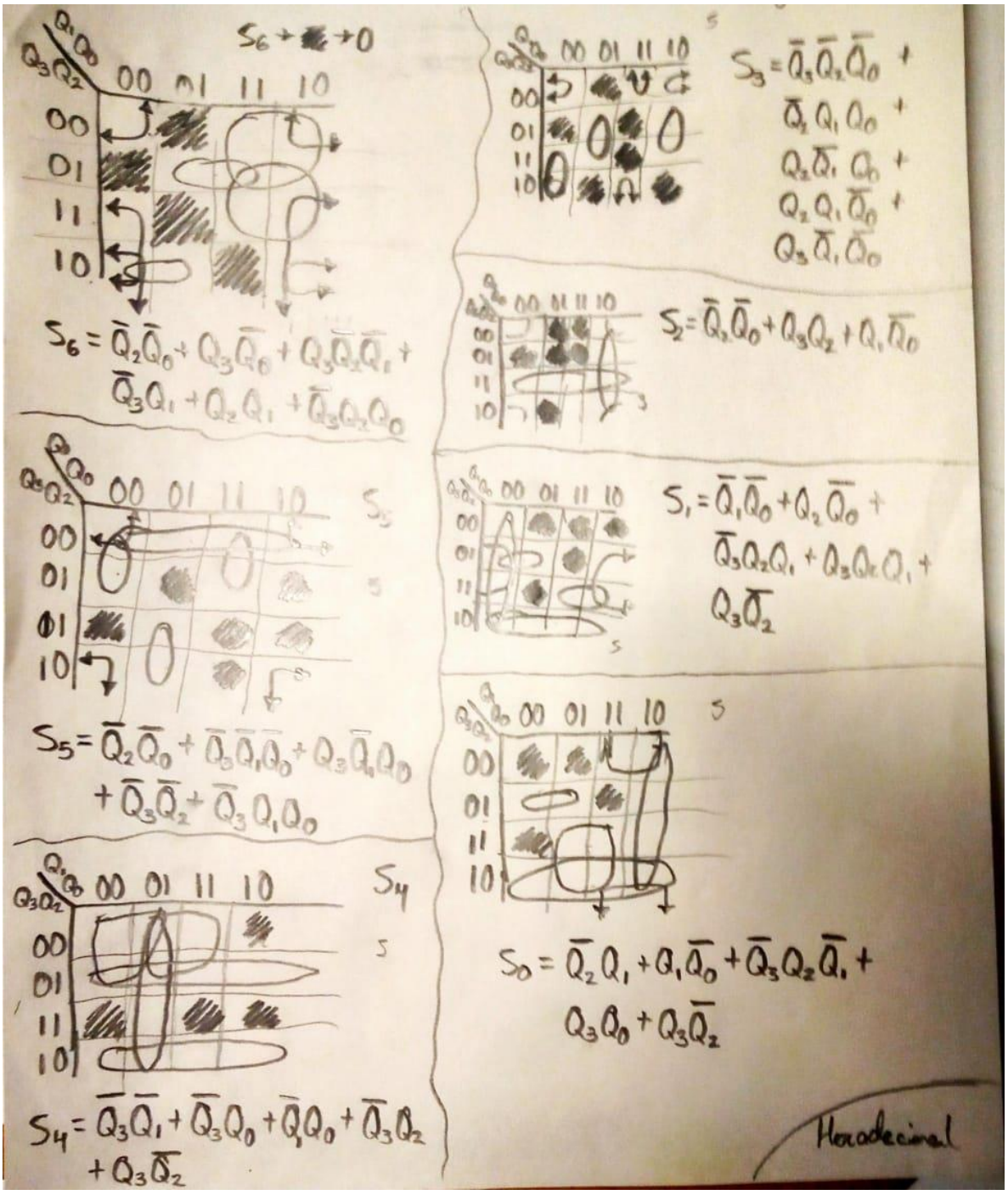
$T_2 = EQ_1Q_0$

$T_1 = EQ_0$

$T_0 = E$

Estas salen por análisis visual

Sólo los T_i dependen de E, por lo que todas las S son con mapas de Karnaugh de 4 variables


S6 + ■ + 0
Q1Q0
Q3Q2
00 01 11 10
00
01
11
10
S6 = Q̄2Q̄0 + Q3Q̄0 + Q3Q̄2Q̄1 +
Q̄3Q1 + Q2Q1 + Q̄3Q2Q0
S3 = Q̄3Q̄2Q̄0 +
Q̄2Q1Q0 +
Q2Q̄1Q0 +
Q2Q1Q̄0 +
Q3Q̄1Q̄0
S2 = Q̄2Q̄0 + Q3Q2 + Q1Q̄0
S1 = Q̄1Q̄0 + Q2Q̄0 +
Q̄3Q2Q1 + Q3Q2Q1 +
Q3Q̄2
S5 = Q̄2Q̄0 + Q̄3Q̄1Q̄0 + Q3Q̄1Q0
+ Q̄3Q̄2 + Q̄3Q1Q0
S4 = Q̄3Q̄1 + Q̄3Q0 + Q̄1Q0 + Q̄3Q2
+ Q3Q̄2
S0 = Q̄2Q1 + Q1Q̄0 + Q̄3Q2Q̄1 +
Q3Q0 + Q3Q̄2
Hexadecimal

NOMBRE

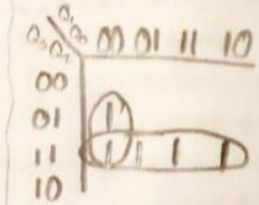
Mensaje Brayan Yosafat

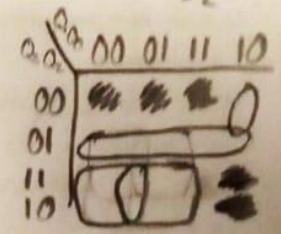
E	$Q_3Q_2Q_1Q_0$	$D_3D_2D_1D_0$	S	S_6	S_5	S_4	S_3	S_2	S_1	S_0
1	0000	0001	b	0	0	1	1	1	1	1
1	0001	0011	r	0	0	0	0	1	0	1
1	0011	0010	A	1	1	1	0	1	1	1
1	0010	0110	y	0	1	1	1	0	1	1
1	0110	0111	A	1	1	1	0	1	1	1
1	0111	0101	n	0	0	1	0	1	0	1
1	0101	0100	y	0	1	1	1	0	1	1
1	0100	1100	□	1	1	1	1	1	1	0
1	1100	1101	S	1	0	1	1	0	1	1
1	1101	1111	A	1	1	1	0	1	1	1
1	1111	1110	F	1	0	0	0	1	1	1
1	1110	1010	A	1	1	1	0	1	1	1
1	1010	0000	t	0	0	0	1	1	1	1

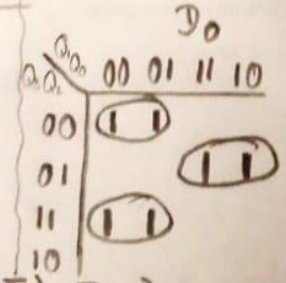
La forma más fácil es el resultado del mapa por $\bar{E}$

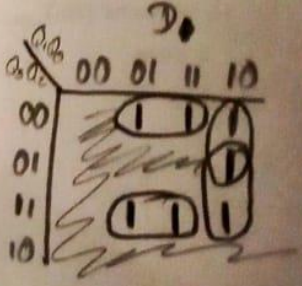
$D_1 = E\,mapa + \bar{E}Q_1$

Si E=0 guarda mapa y pasa Q_1







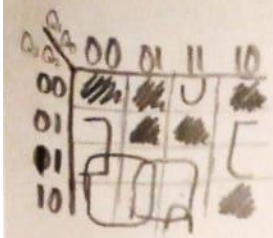


$D_3 = E\left(Q_2\bar{Q}_1\bar{Q}_0 + Q_3Q_2\right) + \bar{E}Q_3$

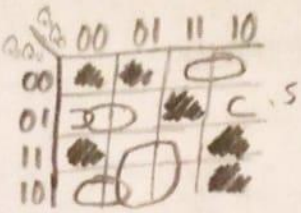
$D_2 = E\left(\bar{Q}_3Q_2 + \bar{Q}_3Q_1\bar{Q}_0 + Q_3\bar{Q}_1 + Q_3Q_0\right) + \bar{E}Q_2$

$D_1 = E\left(\bar{Q}_3\bar{Q}_2Q_0 + Q_3Q_2Q_0 + \bar{Q}_3Q_1\bar{Q}_0 + Q_2Q_1\bar{Q}_0\right) + \bar{E}Q_1$

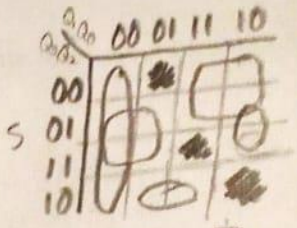
$D_0 = E\left(\bar{Q}_3\bar{Q}_2\bar{Q}_1 + \bar{Q}_3Q_2Q_1 + Q_3Q_2\bar{Q}_1\right) + \bar{E}Q_0$



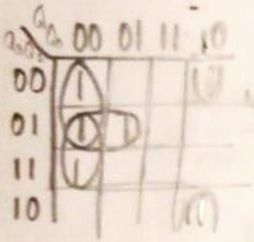
$S_6 = \bar{Q}_2 Q_1 Q_0 + Q_2 \bar{Q}_0 + Q_3 \bar{Q}_1 + Q_3 Q_0$



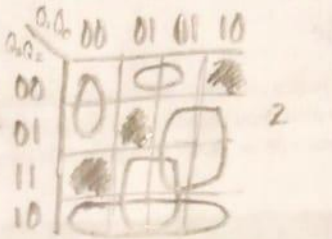
$S_5 = \bar{Q}_3 \bar{Q}_2 Q_1 + \bar{Q}_3 Q_2 \bar{Q}_0 + \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_0 + Q_3 \bar{Q}_2 \bar{Q}_1$



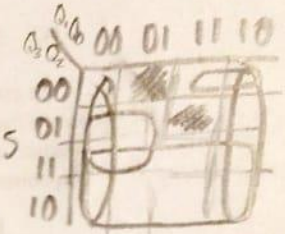
$S_4 = \bar{Q}_1 \bar{Q}_0 + \bar{Q}_3 Q_1 + Q_2 \bar{Q}_1 + Q_2 Q_1 \bar{Q}_0 + Q_3 \bar{Q}_2 Q_0$



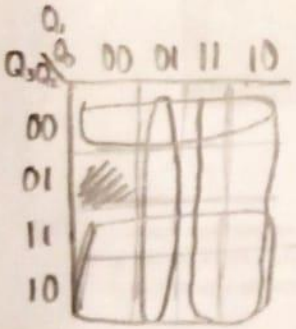
$S_3 = \bar{Q}_3 \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + Q_2 \bar{Q}_1 \bar{Q}_0 + \bar{Q}_3 Q_2 \bar{Q}_1 + \bar{Q}_2 Q_1 \bar{Q}_0$



$S_2 = \bar{Q}_3 \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + \bar{Q}_3 \bar{Q}_2 Q_1 \bar{Q}_0 + Q_2 \bar{Q}_1 + Q_3 Q_0 + Q_3 \bar{Q}_2$

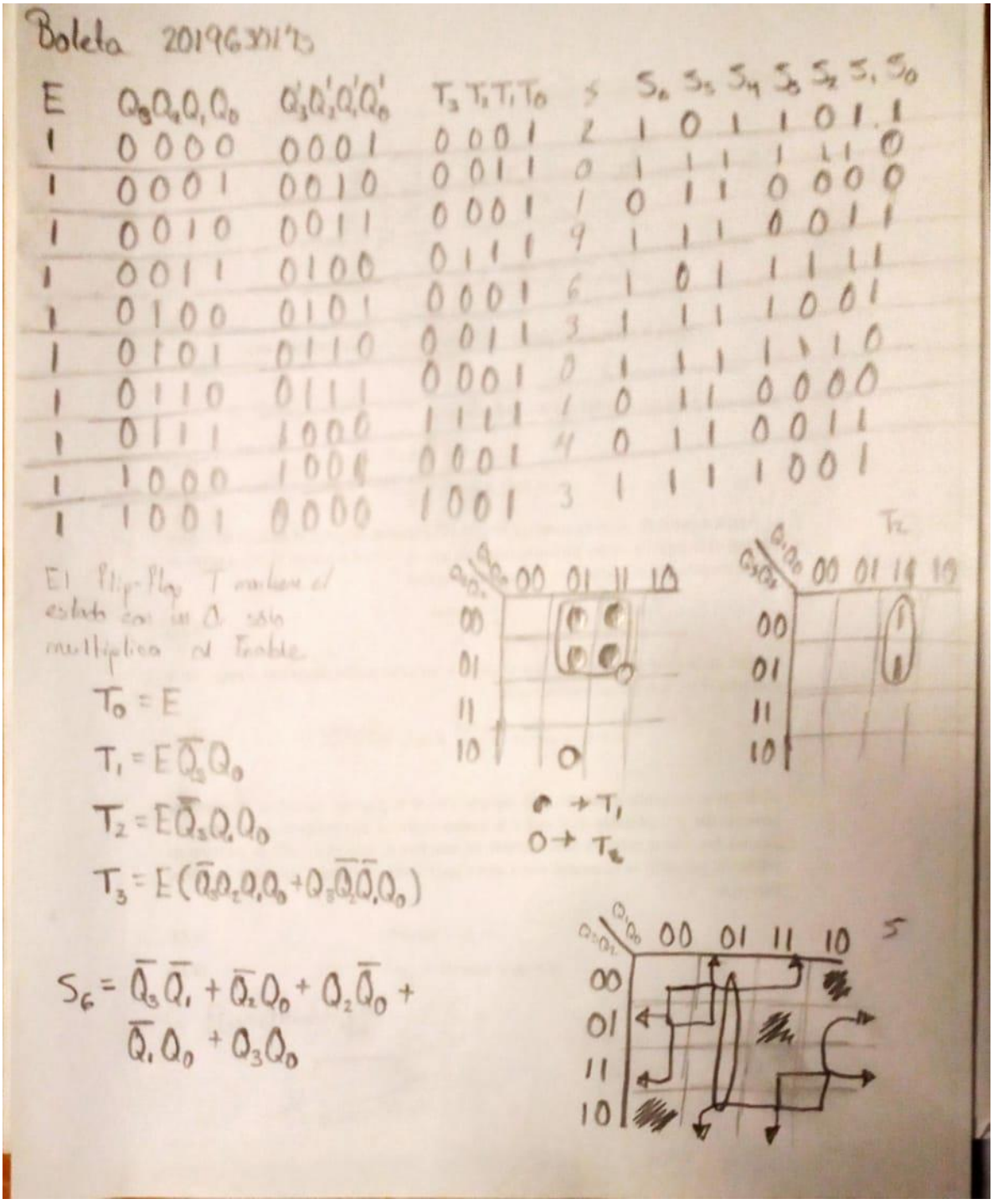


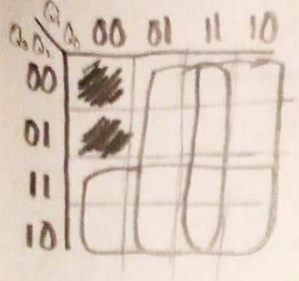
$S_1 = \bar{Q}_1 \bar{Q}_0 + \bar{Q}_3 \bar{Q}_2 Q_1 + Q_1 \bar{Q}_0 + Q_2 \bar{Q}_1 + Q_3$



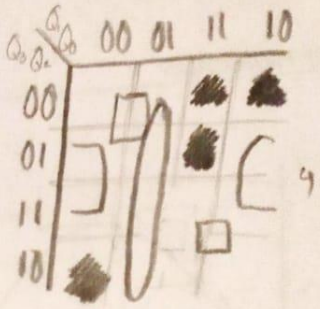
$S_0 = \bar{Q}_3 \bar{Q}_2 + \bar{Q}_1 Q_0 + Q_1 + Q_3$

BOLETA


Boleta 20196301'23
E $Q_3Q_2Q_1Q_0$ $Q_3'Q_2'Q_1'Q_0'$ $T_3T_2T_1T_0$
El Flip-Flop T cambia el estado con un 0, esto multiplica el Teorema
$T_0 = E$
$T_1 = E\bar{Q}_3Q_0$
$T_2 = E\bar{Q}_3Q_1Q_0$
$T_3 = E(\bar{Q}_3Q_2Q_1Q_0 + Q_3\bar{Q}_2\bar{Q}_1Q_0)$
$S_6 = \bar{Q}_3\bar{Q}_1 + \bar{Q}_2Q_0 + Q_2\bar{Q}_0 + \bar{Q}_1Q_0 + Q_3Q_0$
1 → T_1
0 → T_0

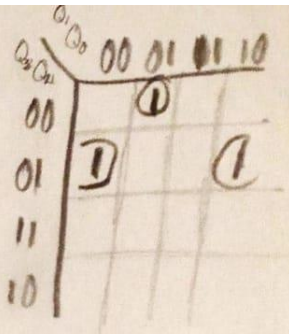


$S_5 = Q_3 + Q_0 + Q_1$

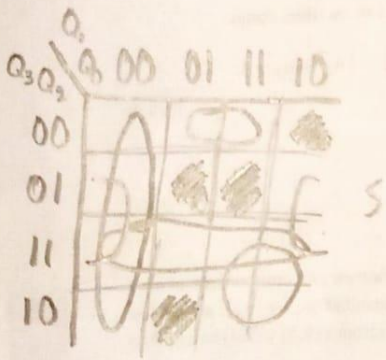


$S_4 = 1$

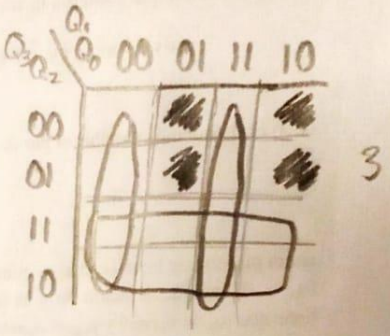
$S_3 = \bar{Q}_3\bar{Q}_1 + \bar{Q}_1Q_0 + Q_2\bar{Q}_0 + Q_3Q_1$



$S_2 = \bar{Q}_3\bar{Q}_2\bar{Q}_1Q_0 + \bar{Q}_3Q_2\bar{Q}_0$



$S_1 = \bar{Q}_3\bar{Q}_2Q_0 + \bar{Q}_1\bar{Q}_0 + Q_2\bar{Q}_0 + Q_3Q_2 + Q_3Q_1$



$S_0 = \bar{Q}_1\bar{Q}_0 + Q_1Q_0 + Q_3$

CÓDIGO FUENTE

DADO

```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3
 4 entity Dado is port (
 5     CLK, CLR, E : in std_logic;
 6     S : out std_logic_vector (6 downto 0)
 7 );
 8 end entity;
 9
10 architecture aDado of Dado is
11 signal FF : std_logic_vector (6 downto 0);
```

```
12 begin
13     process (CLK, CLR)
14     begin
15         if CLR = '1' then
16             FF <= "0000000";
17         elsif rising_edge(CLK) then
18             if E = '0' then
19                 FF <= FF;
20             else
21                 case FF is
22                     when "0110000" => --Es un 1
23                         FF <= "1101101";
24                     when "1101101" => --Es un 2
25                         FF <= "1111001";
26                     when "1111001" => --Es un 3
27                         FF <= "0110011";
28                     when "0110011" => --Es un 4
29                         FF <= "1011011";
30                     when "1011011" => --Es un 5
31                         FF <= "1011111";
32                     when others => --Es un 6 (u otra cosa)
33                         FF <= "0110000";
34                 end case;
35             end if;
36         end if;
37         S <= FF;
38     end process;
39 end architecture;
```

HEXADECIMAL

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

entity Hexa is port (
    CLK, CLR, E : in std_logic;
    S : out std_logic_vector (6 downto 0)
);
end entity;

architecture aHexa of Hexa is
signal FF : std_logic_vector (6 downto 0);
begin
    process (CLK, CLR)
    begin
        if CLR = '1' then
            FF <= "0000000";
        elsif rising_edge(CLK) then
            if E = '0' then
                FF <= FF;
            else
                case FF is
                    when "1111110" => --Es un 0
                        FF <= "0110000";
                    when "0110000" => --Es un 1
                        FF <= "1101101";
                    when "1101101" => --Es un 2
                        FF <= "1111001";
                    when "1111001" => --Es un 3
                        FF <= "1011011";
                    when "1011011" => --Es un 5
                        FF <= "1011111";
                    when "1011111" => --Es un 6
                        FF <= "1110000";
                    when "1110000" => --Es un 7
                        FF <= "1111111";
                    when "1111111" => --Es un 8
                        FF <= "1111011";
                    when "1111011" => --Es un 9
                        FF <= "1110111";
                    when "1110111" => --Es una A
                        FF <= "0011111";
                    when "0011111" => --Es una B
                        FF <= "1001110";
                    when "1001110" => --Es una C
                        FF <= "0111101";
                    when "0111101" => --Es una D
                        FF <= "1001111";
                    when "1001111" => --Es una E
                        FF <= "1000111";
                    when others => --Es una F (u otra cosa)
                        FF <= "1111110";
                end case;
            end if;
        end if;
        S <= FF;
    end process;

end architecture;
```

NOMBRE

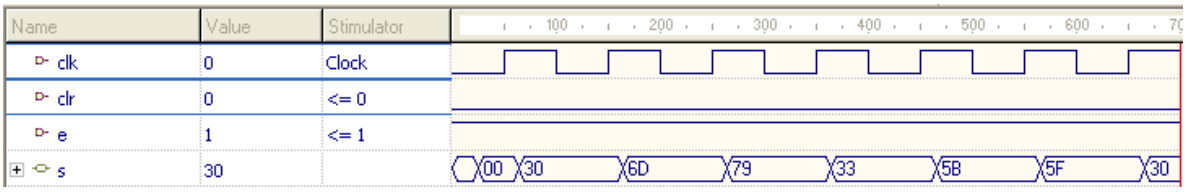
```
library ieee;
use ieee.std_logic_1164.all;

entity Nombre is port (
    CLK, CLR, E : in std_logic;
    S : out std_logic_vector (8 downto 0)
);
end entity;

architecture aNombre of Nombre is
signal FF : std_logic_vector (8 downto 0);
begin
    process (CLK, CLR)
    begin
        if CLR = '1' then
            FF <= "000000000";
        elsif rising_edge(CLK) then
            if E = '0' then
                FF <= FF;
            else
                case FF is
                    when "000011111" => --00B
                        FF <= "000001101";
                    when "000001101" => --00R
                        FF <= "001110111";
                    when "001110111" => --00A
                        FF <= "000111011";
                    when "000111011" => --00Y

                        FF <= "011110111";
                    when "011110111" => --01A
                        FF <= "000010101";
                    when "000010101" => --00N
                        FF <= "010111011";
                    when "010111011" => --01Y
                        FF <= "001111110";
                    when "001111110" => --00O
                        FF <= "001011011";
                    when "001011011" => --00S
                        FF <= "101110111";
                    when "101110111" => --10A
                        FF <= "001000111";
                    when "001000111" => --00F
                        FF <= "111110111";
                    when "111110111" => --11A
                        FF <= "000001111";
                    when others =>    --00T(u otra cosa)
                        FF <= "000011111";
                end case;
            end if;
        end if;
        S <= FF;
    end process;
end architecture;
```

BOLETA

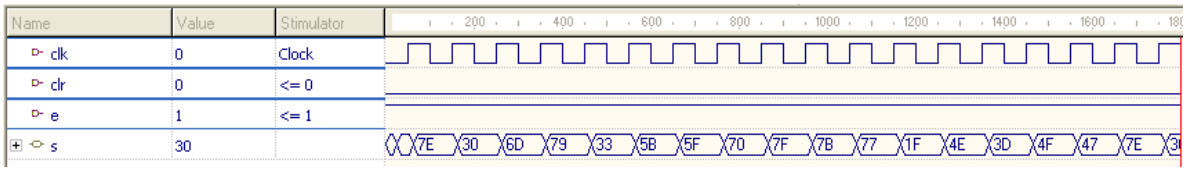
```
library ieee;
use ieee.std_logic_1164.all;

entity Boleta is port (
    CLK, CLR, E : in std_logic;
    S : out std_logic_vector (7 downto 0)
);
end entity;

architecture aBoleta of Boleta is
signal FF : std_logic_vector (7 downto 0);
begin
    process (CLK, CLR)
    begin
        if CLR = '1' then
            FF <= "00000000";
        elsif rising_edge(CLK) then
            if E = '0' then
                FF <= FF;
            else
                case FF is
                    when "01101101" => --02
                        FF <= "01111110";
                    when "01111110" => --00
                        FF <= "00110000";
                    when "00110000" => --01
                        FF <= "01111011";
                    when "01111011" => --09

                        FF <= "01011111";
                    when "01011111" => --06
                        FF <= "01111001";
                    when "01111001" => --03
                        FF <= "11111110";
                    when "11111110" => --10
                        FF <= "10110000";
                    when "10110000" => --11
                        FF <= "00110011";
                    when "00110011" => --04
                        FF <= "11111001";
                    when others =>      --13(u otra cosa)
                        FF <= "01101101";
                end case;
            end if;
        end if;
        S <= FF;
    end process;
end architecture;
```

SIMULACIONES EN GALAXY

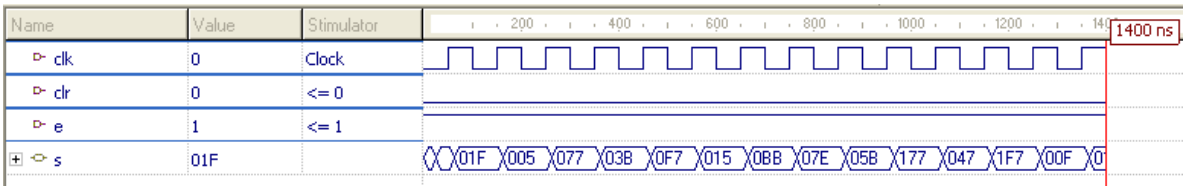
DADO



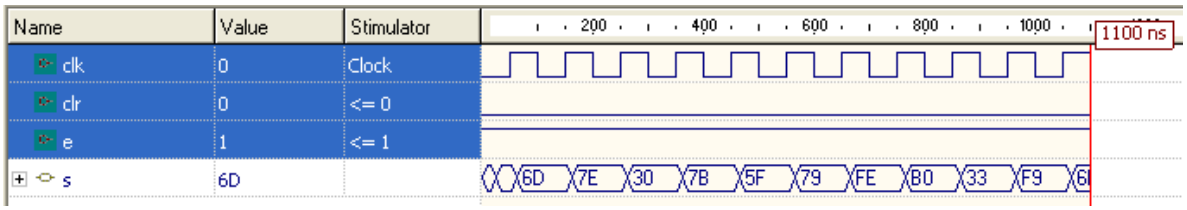
HEXADECIMAL



NOMBRE



BOLETA

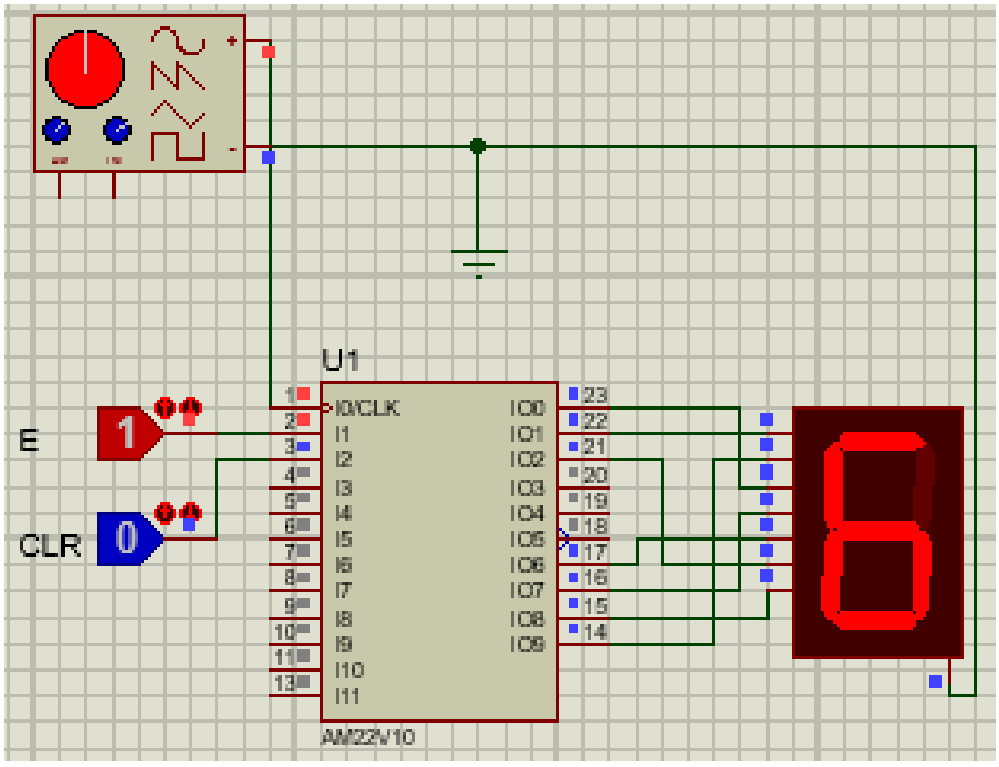


SIMULACIÓN EN PROTEUS

DADO

C22V10

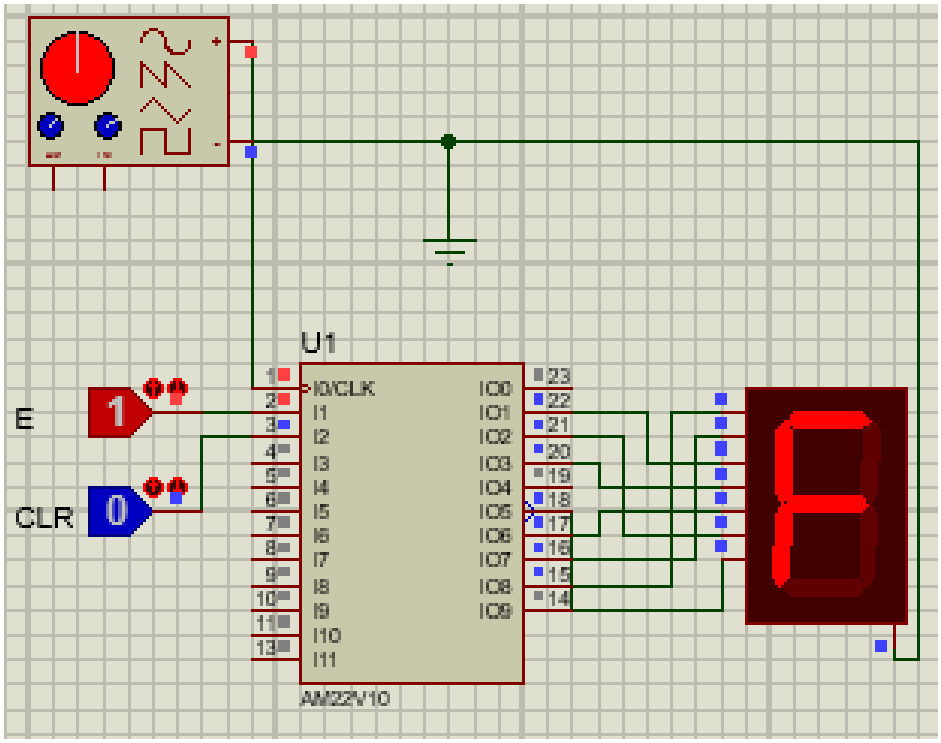
```
                                                          |24|* not used
      clk =|  1|                                         |23|= s(4)
        e =|  2|                                         |22|= s(6)
      clr =|  3|                                         |21|= s(1)
not used *|  4|                                         |20|* not used
not used *|  5|                                         |19|* not used
not used *|  6|                                         |18|* not used
not used *|  7|                                         |17|= s(2)
not used *|  8|                                         |16|= s(3)
not used *|  9|                                         |15|= s(0)
not used *|10|                                          |14|= s(5)
not used *|11|                                          |13|* not used
not used *|12|
```



HEXADECIMAL

C22V10

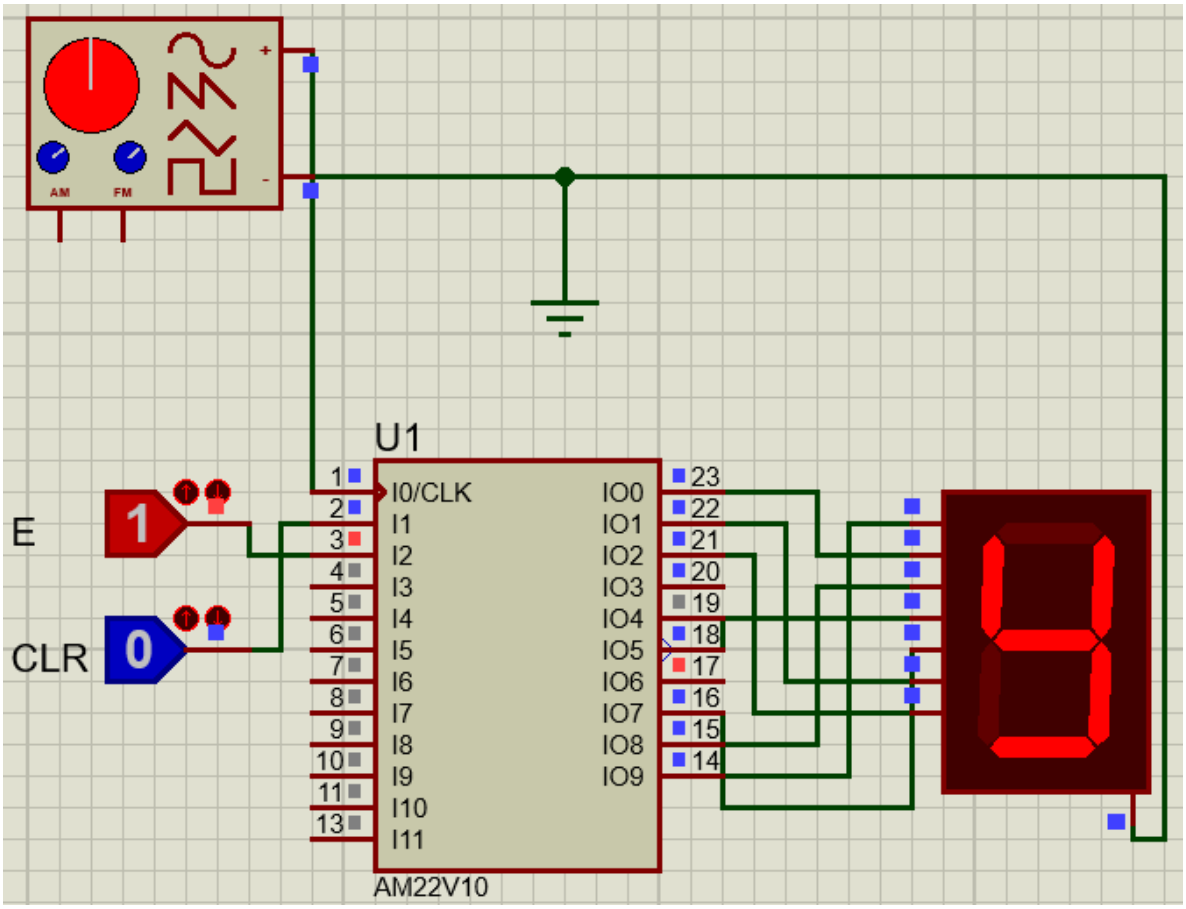
```
                 ____________________________________________
      clk =|  1|                                  |24|* not used
        e =|  2|                                  |23|* not used
      clr =|  3|                                  |22|= s(4)
not used *|  4|                                  |21|= s(1)
not used *|  5|                                  |20|= s(3)
not used *|  6|                                  |19|* not used
not used *|  7|                                  |18|= s(0)
not used *|  8|                                  |17|= s(2)
not used *|  9|                                  |16|= s(5)
not used *|10|                                   |15|= s(6)
not used *|11|                                   |14|* not used
not used *|12|                                   |13|* not used
                 ____________________________________________
```



NOMBRE

C22V10

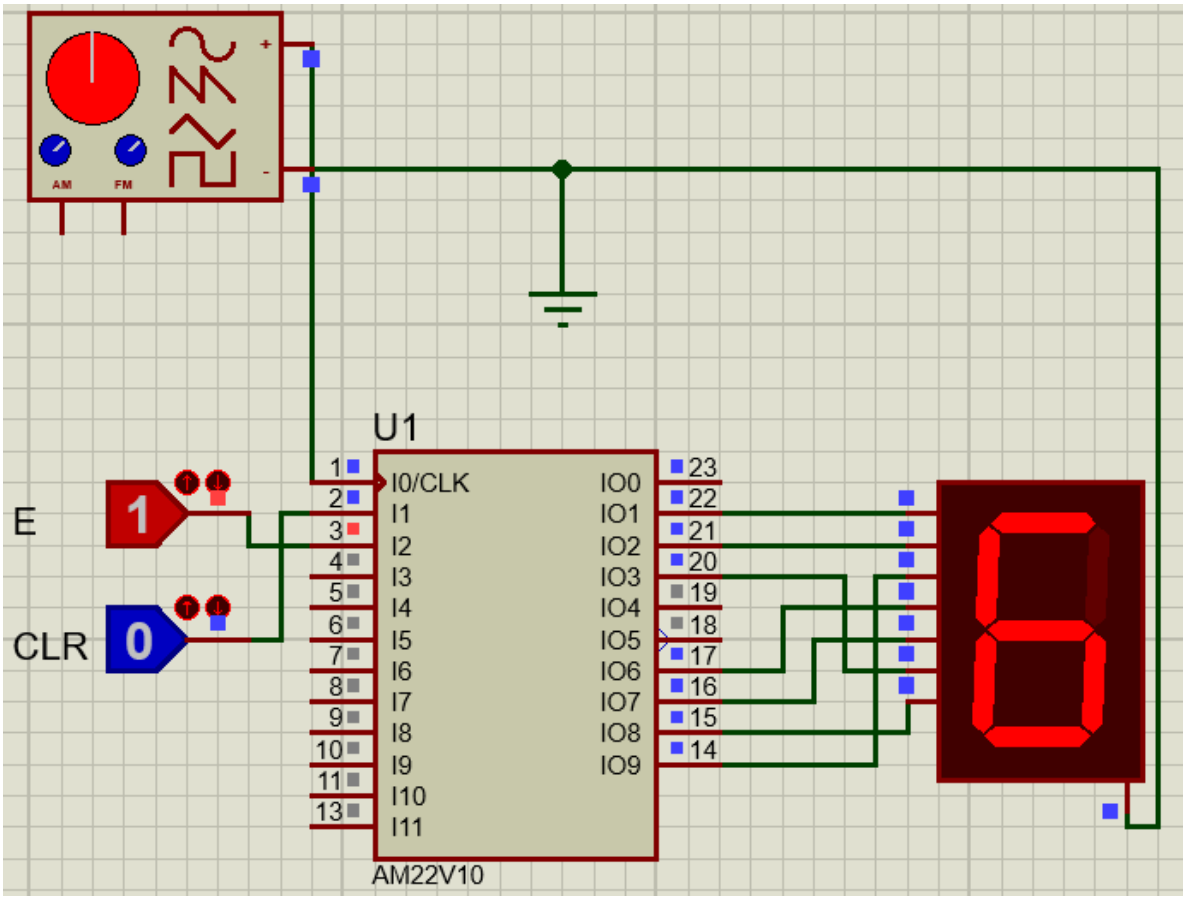
```
                    ____________________________________
        clk =|  1|                                |24|* not used
        clr =|  2|                                |23|= s(5)
          e =|  3|                                |22|= s(1)
not used *|  4|                                |21|= s(0)
not used *|  5|                                |20|= s(8)
not used *|  6|                                |19|* not used
not used *|  7|                                |18|= s(3)
not used *|  8|                                |17|= s(7)
not used *|  9|                                |16|= s(2)
not used *|10|                                |15|= s(4)
not used *|11|                                |14|= s(6)
not used *|12|                                |13|* not used
                    ____________________________________
```



BOLETA

C22V10

```
                 ____________________________________
      clk =|  1|                                |24|* not used
      clr =|  2|                                |23|= s(7)
        e =|  3|                                |22|= s(6)
not used *|  4|                                |21|= s(5)
not used *|  5|                                |20|= s(1)
not used *|  6|                                |19|* not used
not used *|  7|                                |18|* not used
not used *|  8|                                |17|= s(3)
not used *|  9|                                |16|= s(2)
not used *|10|                                 |15|= s(0)
not used *|11|                                 |14|= s(4)
not used *|12|                                 |13|* not used
                 ____________________________________
```



CUESTIONARIO

1. ¿CUÁNTOS DISPOSITIVOS PLD 22V10 SON NECESARIOS PARA EL DESARROLLO DE ESTA PRÁCTICA?

Dado: 1

Hexadecimal: 1

Nombre: 1

Boleta: 1

2. ¿CUÁNTOS DISPOSITIVOS DE LA SERIE 74XX (TTL) Ó 40XX (CMOS) HUBIERAS NECESITADO PARA EL DESARROLLO DE ESTA PRÁCTICA?

Dado: 10

Hexadecimal: 18

Nombre: 19

Boleta: 16

3. ¿CUÁNTOS PINES DE ENTRADA/SALIDA DEL PLD 22V10 SE USAN EN LOS DISEÑOS?

Dado: 10 pines, 45% del total.

Hexadecimal: 10, 45% del total.

Nombre: 12, 54% del total.

Boleta: 11, 50% del total.

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?

Dado: En total, 39, que es el 42% del total.

Hexadecimal: En total 74, el 61% del total.

Nombre: En total 77, 63% del total.

Boleta: En total 64, 52% del total.

5. ¿ES POSIBLE IMPLEMENTAR LOS DISEÑOS USANDO CUALQUIER TIPO DE CODIFICACIÓN EN EL PLD22V10?

No realmente, aunque tiene limitaciones, es bastante potente un solo PLD.

6. ¿CUÁLES SON LAS SEÑALES QUE FUNCIONAN DE MANERA SÍNCRONA Y CUÁLES DE MANERA ASÍNCRONA?

Dado: CLR y CLK son asíncronos, E es síncrona.

Hexadecimal: CLR y CLK son asíncronos, E es síncrona.

Nombre: CLR y CLK son asíncronos, E es síncrona.

Boleta: CLR y CLK son asíncronos, E es síncrona.

7. ¿QUÉ PUEDES CONCLUIR DE ESTA PRÁCTICA?

Como lo suponía desde la práctica pasada, estos son los contadores que generan los números pseudoaleatorios en las computadoras, deben tener todavía más utilidades de las que podemos ver en esta práctica.