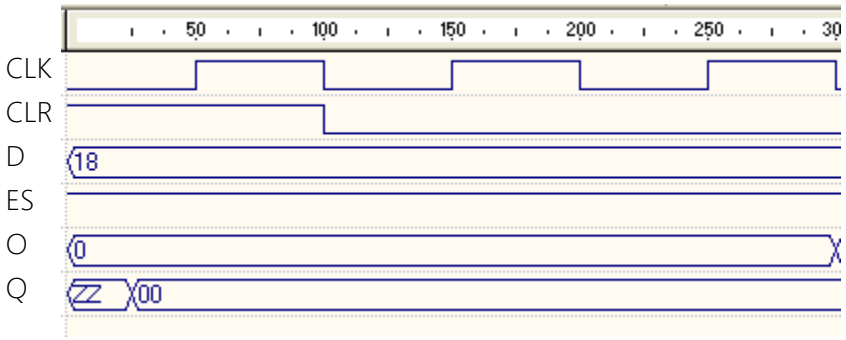
Martínez Coronel Brayan Yosafat

CÓDIGO FUENTE

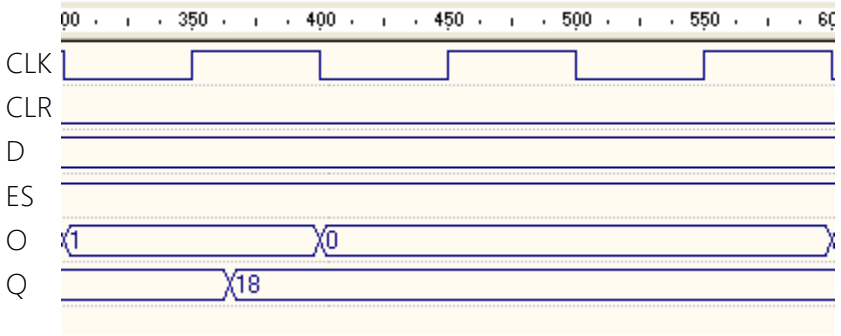
```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3
 4 entity P3 is port(
 5     D : in std_logic_vector (6 downto 0);
 6     O : in std_logic_vector (1 downto 0);
 7     ES, CLK, CLR : in std_logic;
 8     Q : out std_logic_vector (6 downto 0)
 9 );
10 end P3;
11
12 architecture AP3 of P3 is
13 signal aQ, aD : std_logic_vector (6 downto 0);
14 begin
15     --Multiplexor
16     process (O, D, aQ, ES, aD) begin
17         case O is
18             when "00" => aD <= aQ;--Retencion
19             when "01" => aD <= D;--Carga
20             when "10" => --C. a la izq
21                 for i in 1 to 6 loop
22                     aD(i) <= aQ(i - 1);
23                 end loop;
24
25                 aD(0) <= ES;
26             when others => --C. a la der
27                 for i in 0 to 5 loop
28                     aD(i) <= aQ(i + 1);
29                 end loop;
30
31                 aD(6) <= ES;
32         end case;
33     end process;
34
35     --Registro
36     process (CLK, CLR) begin
37         if (CLR = '1') then
38             aQ <= "0000000";
39         elsif (rising_edge(CLK)) then
40             aQ <= aD;
41         end if;
42     end process;
43
44     Q <= aQ;
45 end AP3;
```

SIMULACIONES EN GALAXY

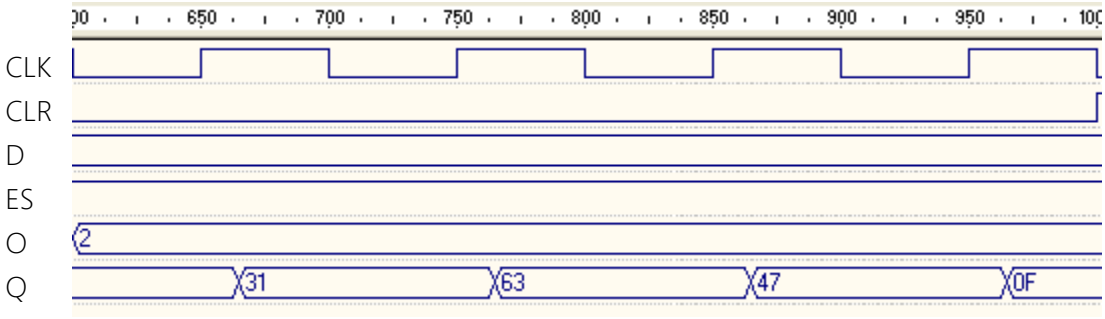
RESET Y RETENCIÓN EN DOS CICLOS DE CLOCK



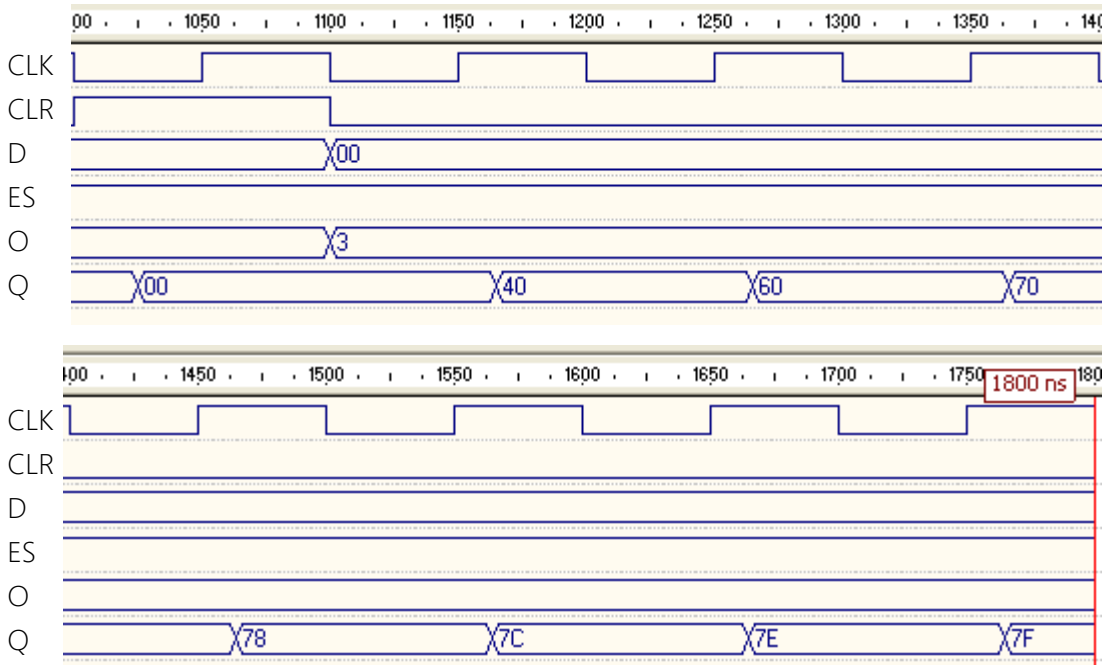
CARGAR 0X18 Y RETENCIÓN DE DOS CICLOS DE CLOCK



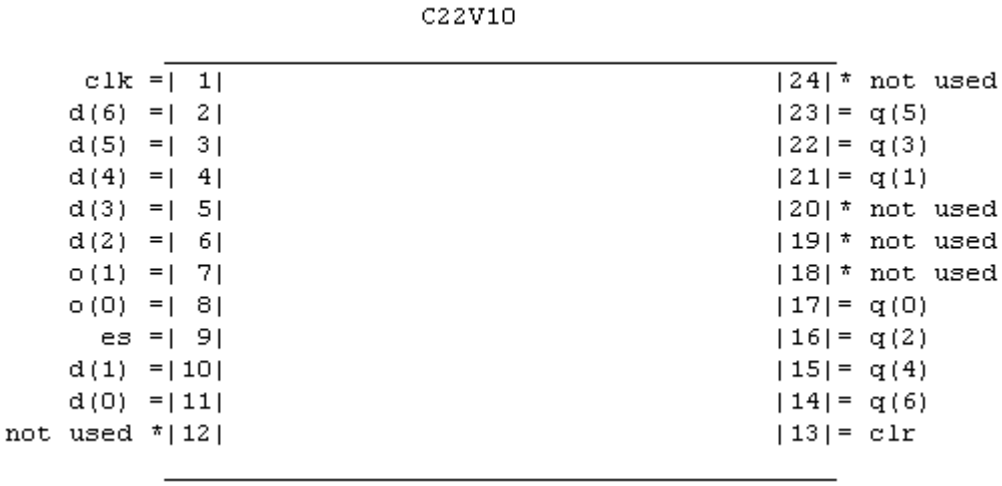
4 CORRIMIENTOS A LAS IZQUIERDA DE 1



RESET Y 7 CORRIMIENTOS A LA DERECHA (SEPARADOS EN 3 Y 4)

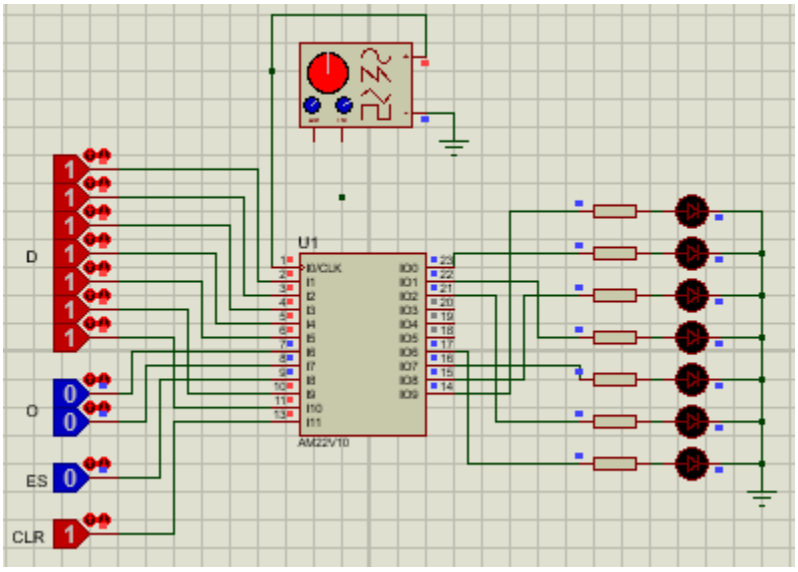


SIMULACIÓN EN PROTEUS

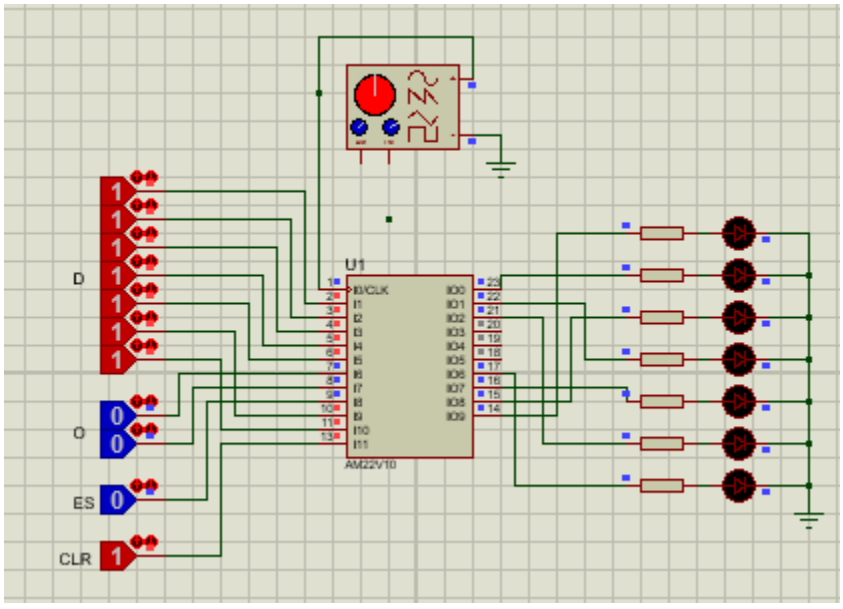


RESET Y RETENCIÓN EN DOS CICLOS DE CLOCK

Reset

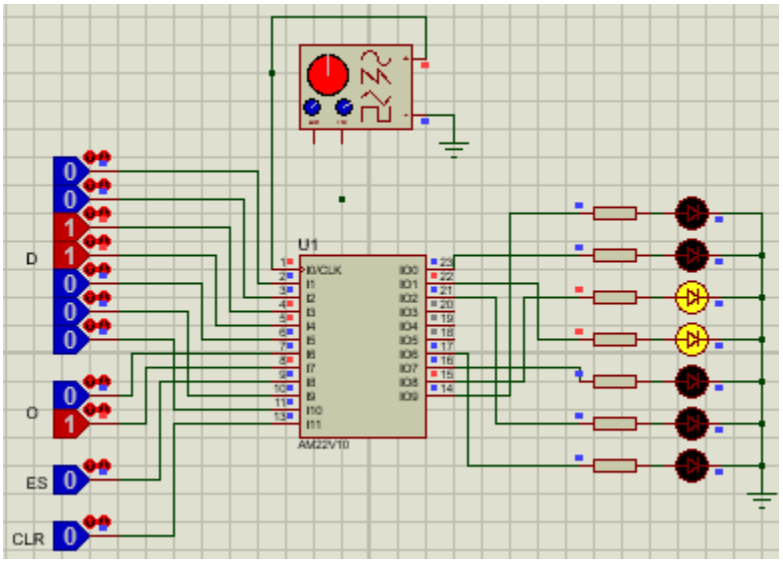


Primer Ciclo (1 segundo a 5)

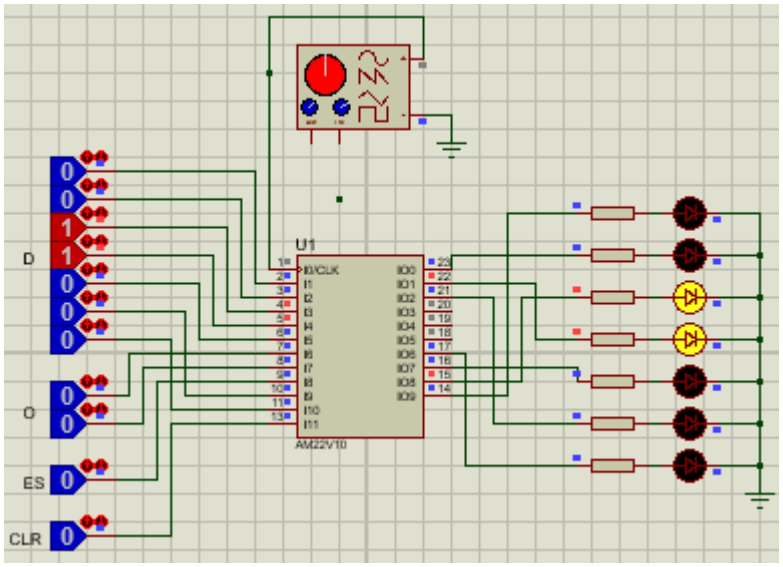


CARGAR 0X18 Y RETENCIÓN DE DOS CICLOS DE CLOCK

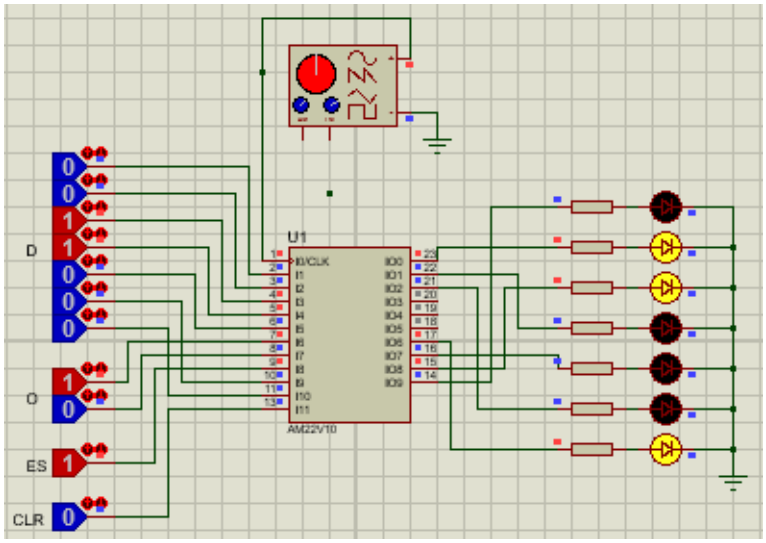
Cargar 0011000

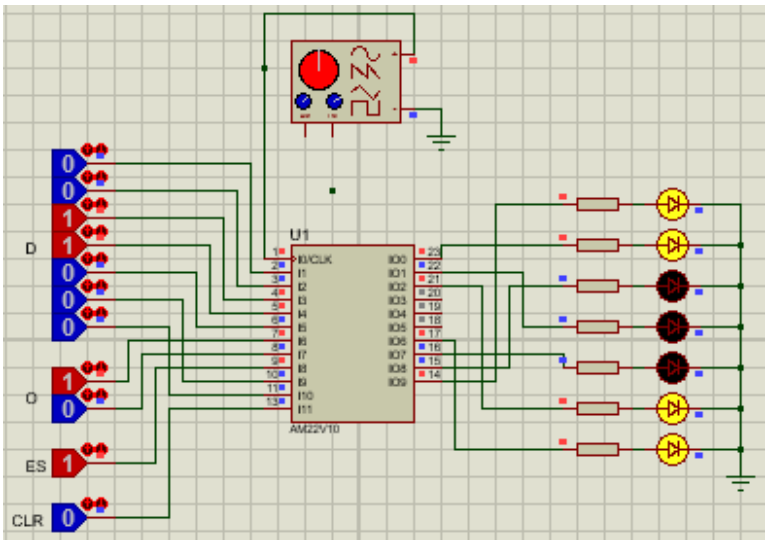


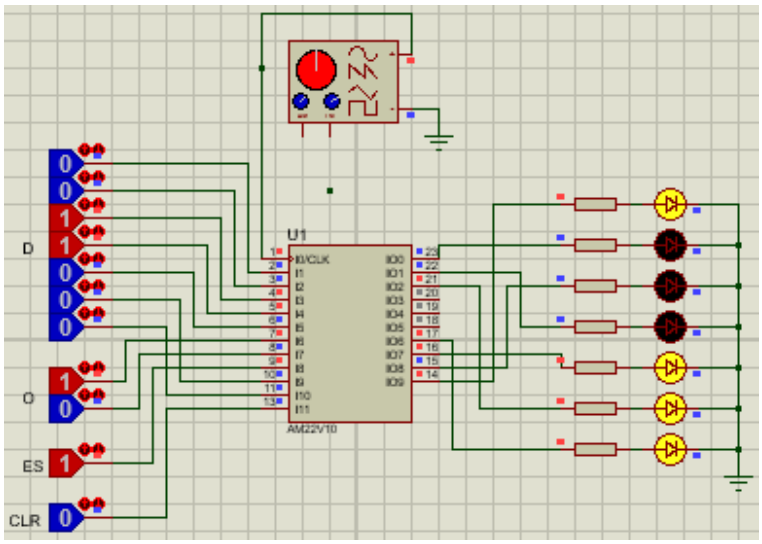
Retención de dos ciclos (Del segundo 6 al 10)

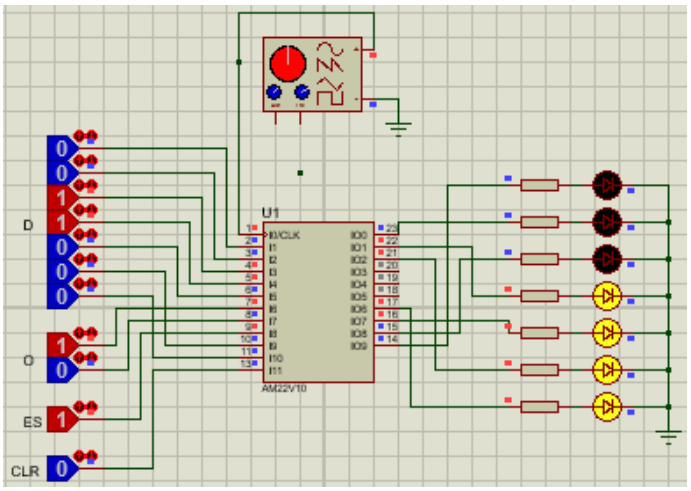


4 CORRIMIENTOS A LAS IZQUIERDA DE 1



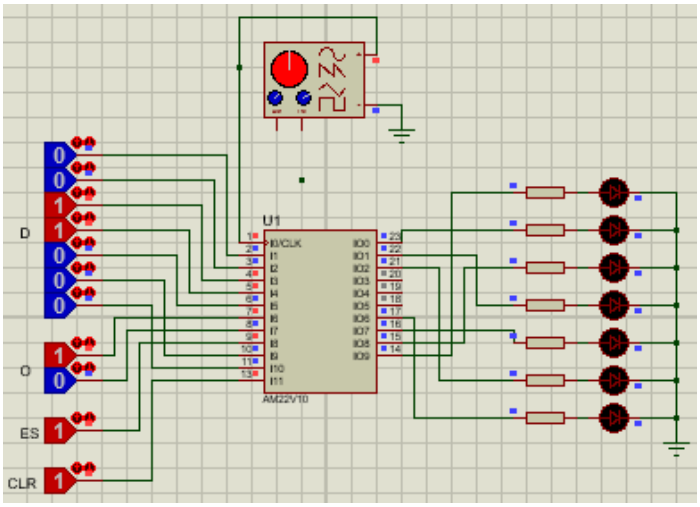




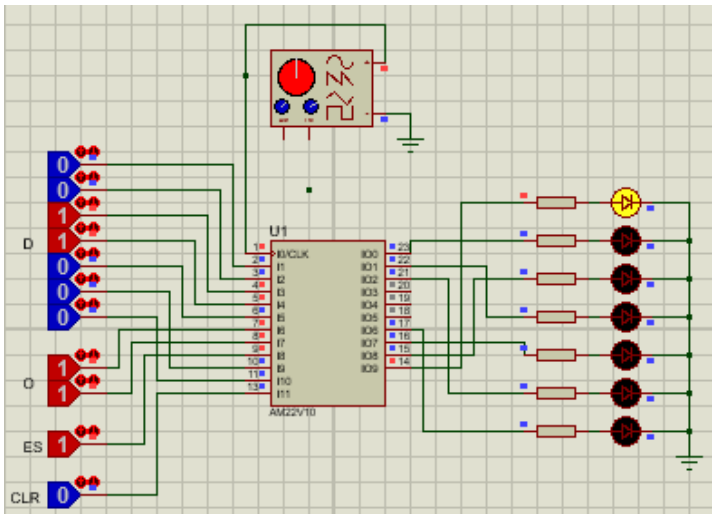


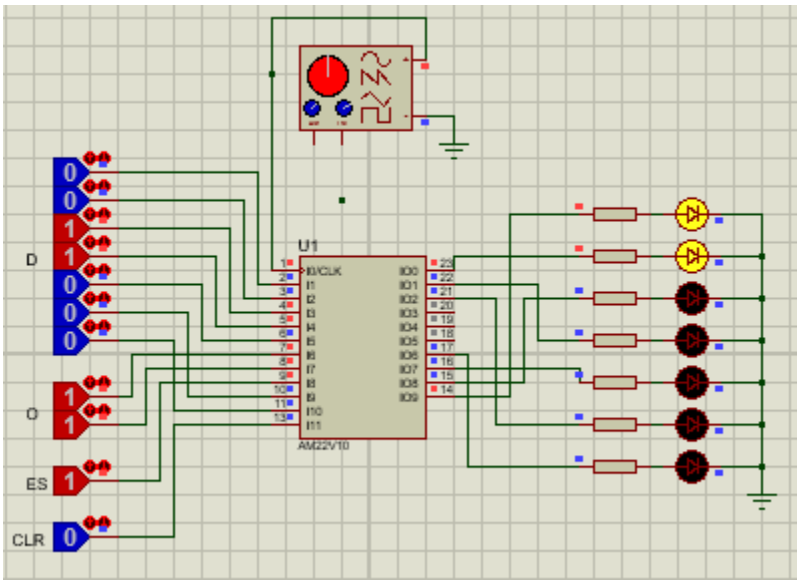
RESET Y 7 CORRIMIENTOS A LA DERECHA (SEPARADOS EN 3 Y 4)

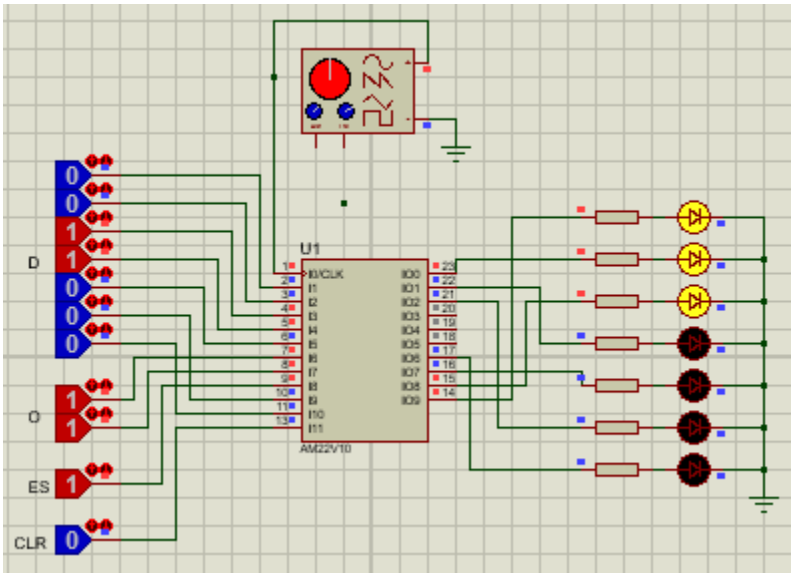
Reset (instantáneo en el frame 6.10)

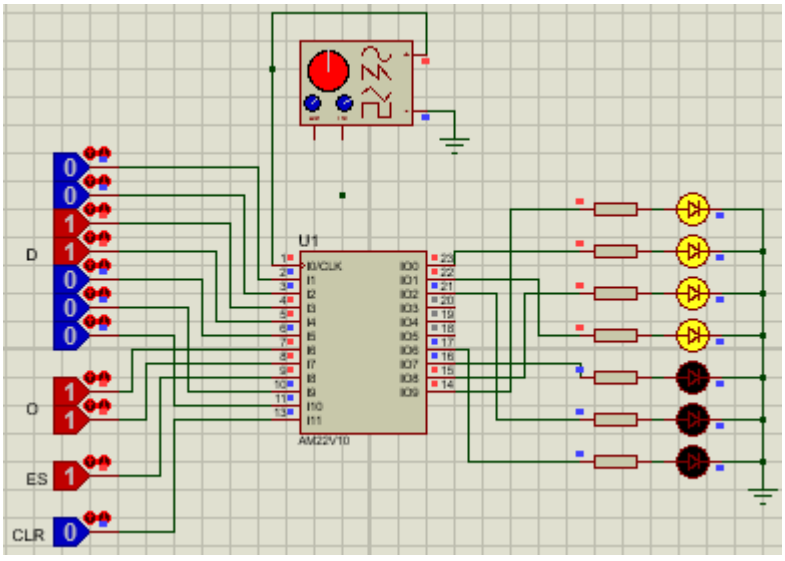


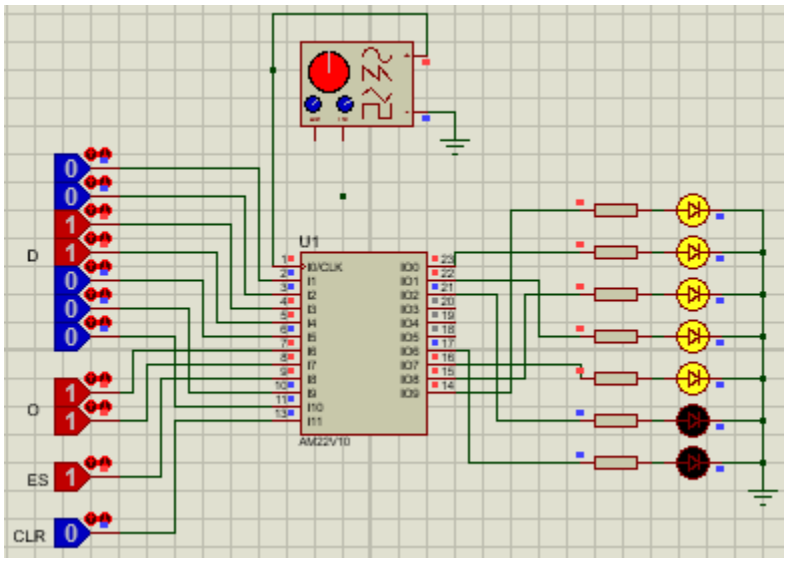
Corrimiento a la derecha de 1

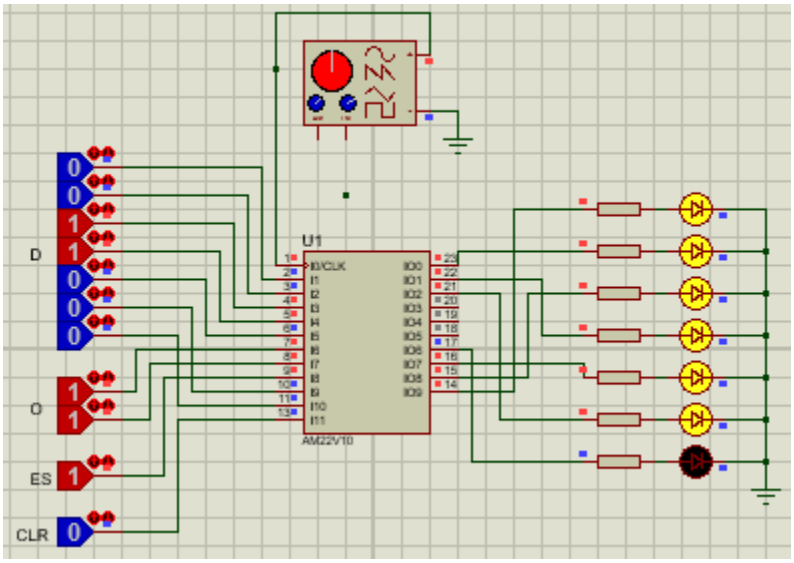


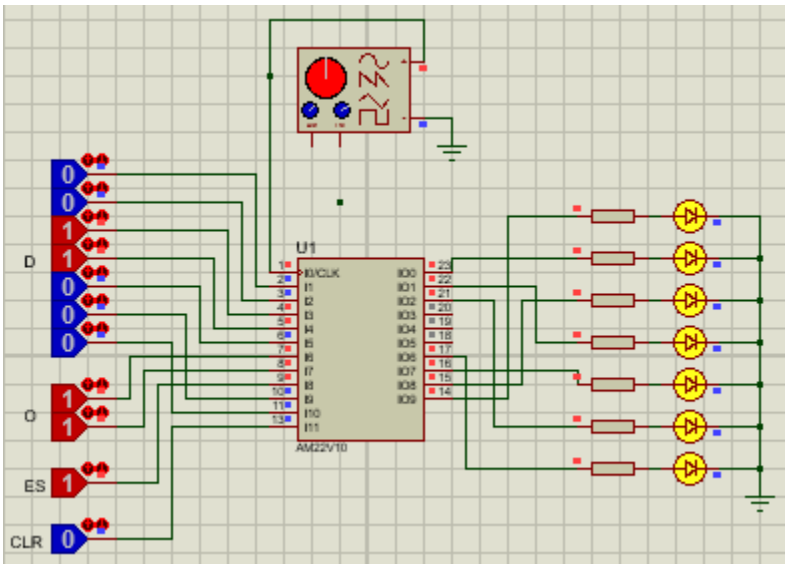

D
O
ES
CLR
U1
AM22V10


D
O
ES
CLR
U1
AM22V10


D
O
ES
CLR
U1
AM22V10


D
O
ES
CLR
U1
AM22V10


D
O
ES
CLR
U1
AM22V10


D
O
ES
CLR
U1
AM22V10

CUESTIONARIO

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

1

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

Considerablemente muchísimos más. Como tendiendo a 20 y con unos cables que parecerían arañas.

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

18 pines en total

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

4 cada una, o sea, 28 en total de los 120. Representa un 23%.

5. ¿Cuáles son tus observaciones con respecto al funcionamiento del registro?

La función del reloj sirve mucho para administrar cómo proteger los datos, de hecho, seguro así es como funcionan las RAM.

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

Asíncronas solo CLR. Todas las demás son síncronas.

7. ¿Qué puedes concluir de esta práctica?

Es bastante funcional para ser relativamente sencillo, si se usa como la memoria RAM entonces lo que lo administre tiene muchas funciones para manipular los datos.