library ieee;

use ieee.std\_logic\_1164.all;

entity practica6A is port(

en, clk, clr : in std\_logic;

q : inout std\_logic\_vector(2 downto 0);

display : out std\_logic\_vector(2 downto 0)

);

end practica6A;

architecture aPractica6A of practica6A is

begin

process(clk, clr)

--usamos variable porque necesitamos cambiar este valor durante el proceso, las señales se actualizan hasta el final del proceso

variable aux : std\_logic;

begin

if (clr = '1') then

q <= (others => '0'); --asigna 0 a todo el vector de q

elsif (rising\_edge(clk)) then

aux := '1'; --se tiene que inicializar aqui, no afuera.

for i in 0 to 2 loop

if (i-1 >= 0) then --para el primer caso donde i es 0, el segundo for entraria en un -1 lo cual causa problemas

for j in 0 to i-1 loop -- este loop es para calcular la parte despues del xor en la formula obtenida

aux := aux and q(j); --por ejemplo si i=5 esto calcula: 1 and q(0), eso lo guarda en aux y a la siguiente calcula aux and q(1), y así hasta terminar...

end loop;

end if;

aux := aux and en; --toma todas la "operaciones and" que calculamos y le hace un and con ENABLE

q(i) <= q(i) xor aux; -- terminamos la formula obtenida añadiendole el término en la parte izq de la operacion xor

end loop;

end if;

end process;

display <= q;

end architecture;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

library ieee;

use ieee.std\_logic\_1164.all;

entity practica6B is port(

en, ud, l, clk, clr : in std\_logic;

c : out std\_logic;

d : in std\_logic\_vector(6 downto 0);

q : inout std\_logic\_vector(6 downto 0) -- utilizaremos esto como nuestra salida

--prescindimos de usar una salida display porque no cabe en una sola GAL

);

end practica6B;

architecture aPractica6B of practica6B is

begin

process(clk, clr)

begin

if (clr='1') then

q <= (others => '0'); --manera elegante de igualar todos los bits de q a 0

elsif (rising\_edge(clk)) then

if (en = '0') then --retencion

q <= q;

elsif (en = '1' and l = '1') then --carga

q <= d;

elsif (en = '1' and l = '0' and ud = '1') then --conteo ascendente aritmetico

q <= q+1;

elsif (en = '1' and l = '0' and ud = '0') then --conteo descendente aritmetico

q <= q-1;

end if;

end if;

if (clr = '0' and (q = "0000000" or q = "1111111")) then

c <= '1';

else

c <= '0';

end if;

end process;

end architecture;

Graphical user interface, application

Description automatically generatedGraphical user interface, application

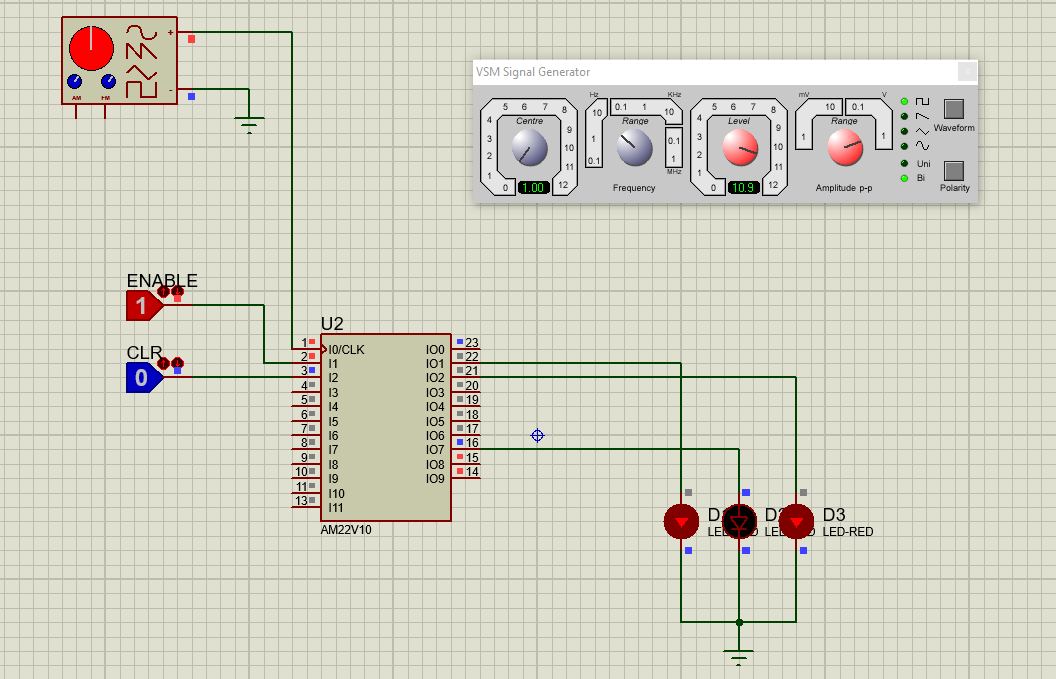
Description automatically generatedGraphical user interface, application

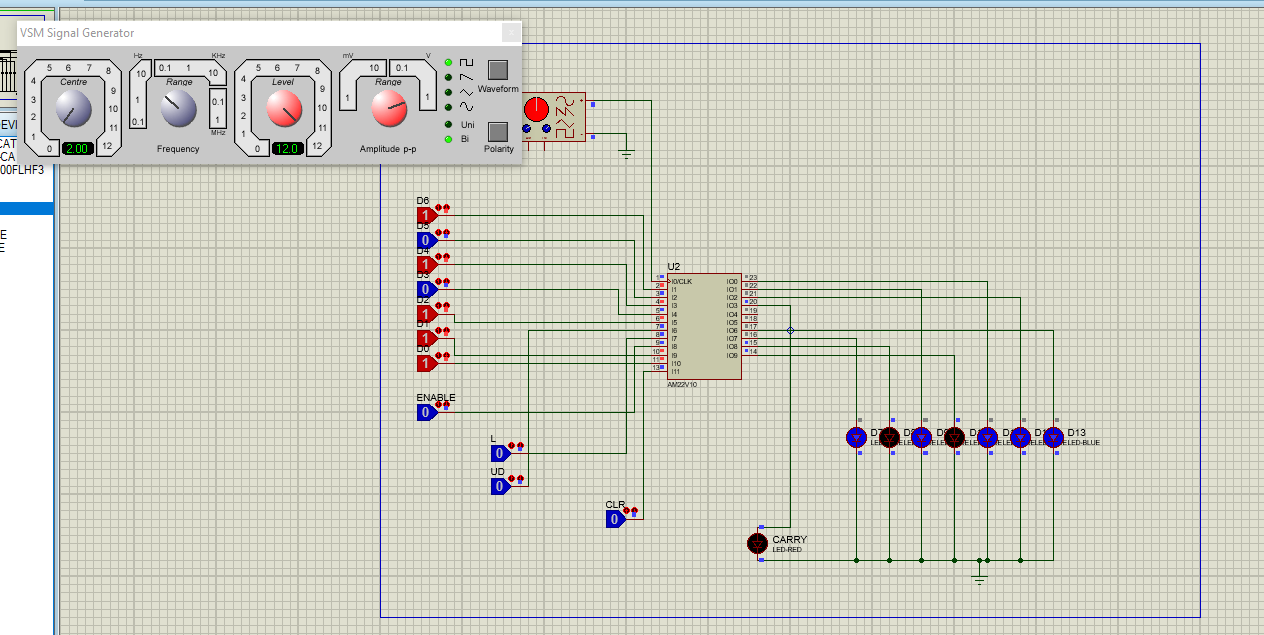
Description automatically generatedGraphical user interface, application

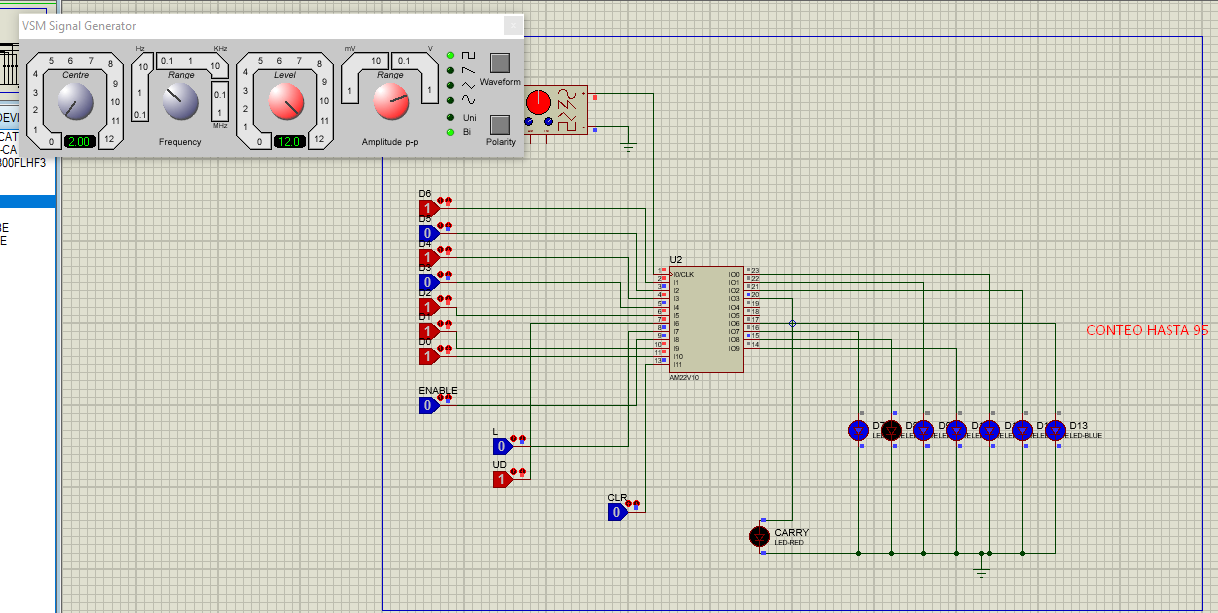
Description automatically generatedGraphical user interface, application

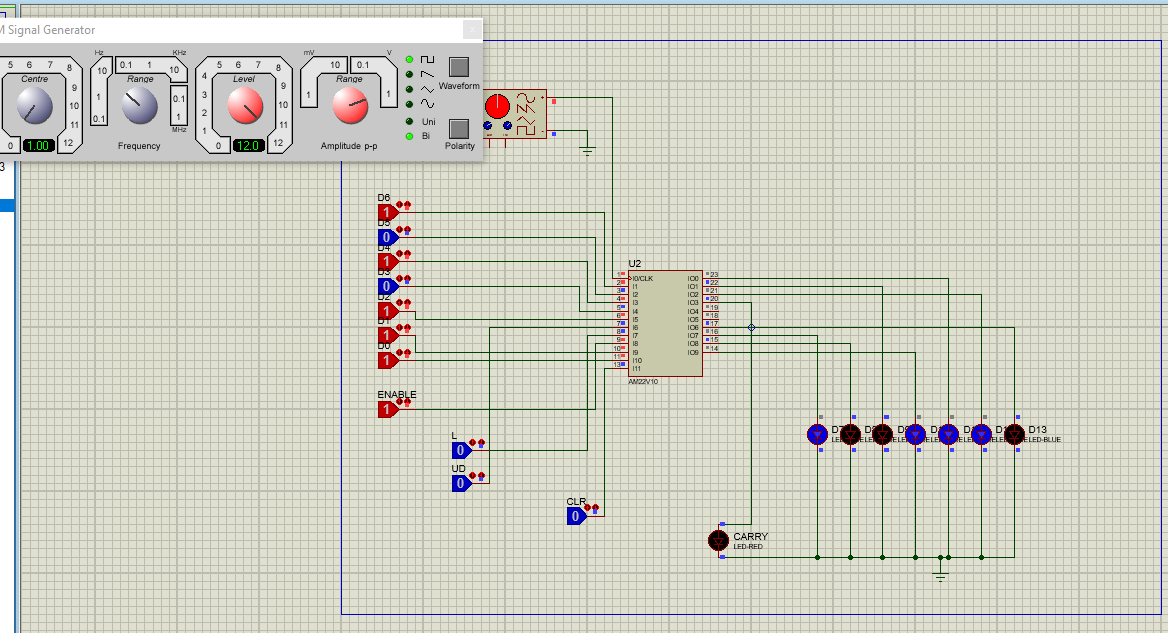
Description automatically generatedGraphical user interface, text, application, email

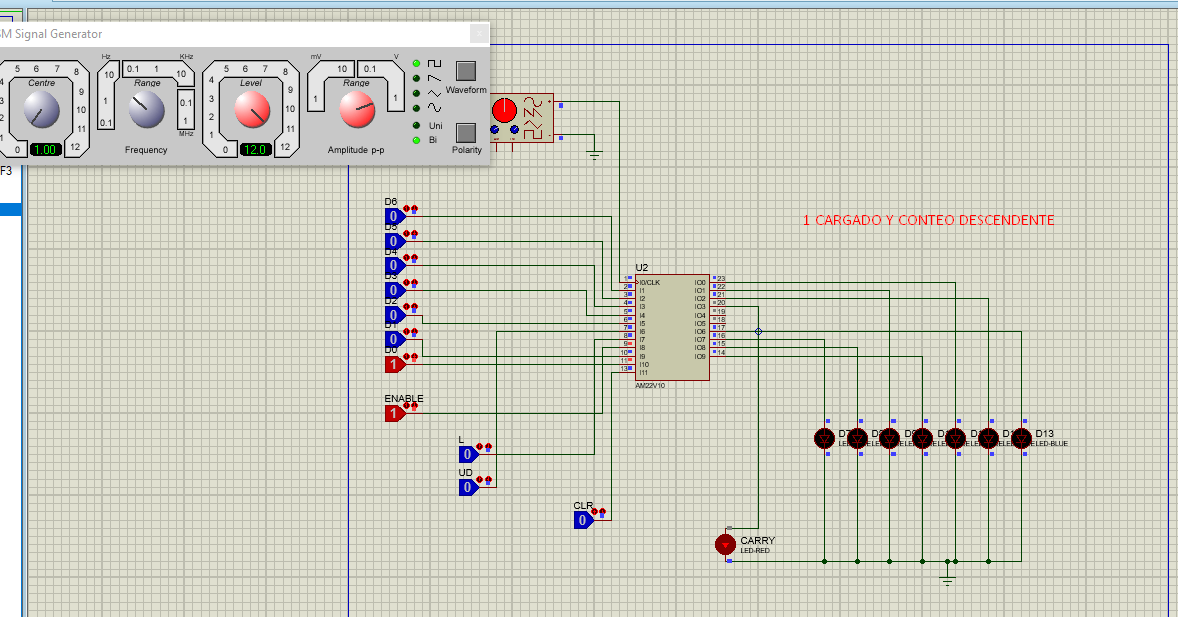
Description automatically generated

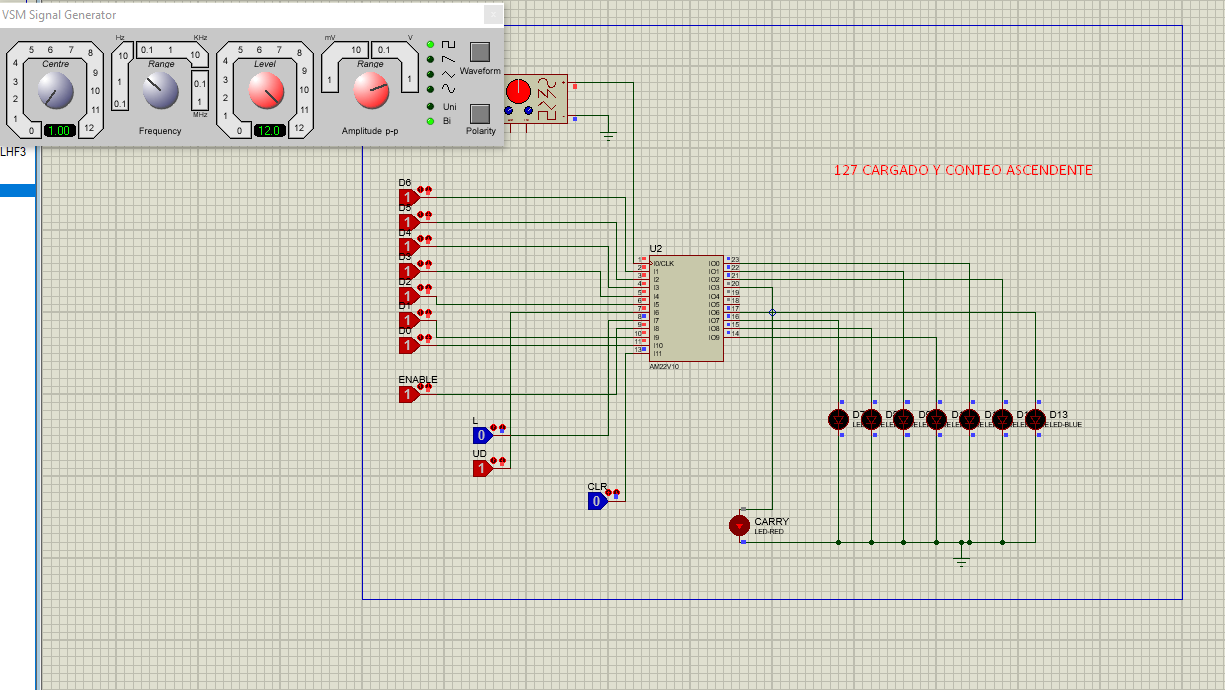












1. ¿Cuántos dispositivos PLD 22V10 son necesarios para el desarrollo de

esta práctica?

**1 para cada contador**

2. ¿Cuántos dispositivos de la serie 74xx (TTL) ó 40xx (CMOS) hubieras

necesitado para el desarrollo de esta práctica?

**18 para el contador de 3 bits, más de 100 para el contador genérico**

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

**3 de entrada y 3 de salida para el de 3 bits**

**12 de entrada y 8 de salida para el contador genérico**

4. ¿Cuántos términos producto ocupan las ecuaciones para cada señal de

salida y que porcentaje se usa en total del PLD 22V10?

**12 términos y 9% de la GAL en el contador de 3 bits**

**56 términos y 46% de la GAL en el contador genérico**

5. ¿Por qué se tienen que usar variables para implementar la ecuación

genérica del contador con señal de control enable?

**Porque las variables pueden cambiar su valor en varias partes del código, pueden cambiar su valor conforme un proceso va avanzando, en cambio, una señal solo puede cambiar su valor una vez, por lo que su valor solo se actualiza al terminar el proceso, como necesitamos manipular esta información dentro del proceso y utilizar sus valores anteriores para calcular el valor nuevo, tenemos que usar una variable en lugar de una señal**

6. ¿Qué nivel de diseño se implementó al usar los operadores + y – en el

contador?

**Diseño aritmético de alto nivel**

7. ¿Cuáles son las señales que funcionan de manera síncrona y cuáles de

manera asíncrona?

**El clr es la señal de control asíncrono, todas las demás son síncronas porque esperan al pulso del reloj**

8. ¿Qué puedes concluir de esta práctica?

**Fue una practica importante porque logramos crear un contador genérico a partir de una fórmula general que nos puede servir para construir cualquier contador genérico de n bits, también utilizamos una librería nueva para facilitarnos la implementación de un contador ascendente y descendente, de esta manera podemos darnos cuenta de las ventajas que tiene la descripción de software en VHDL.**