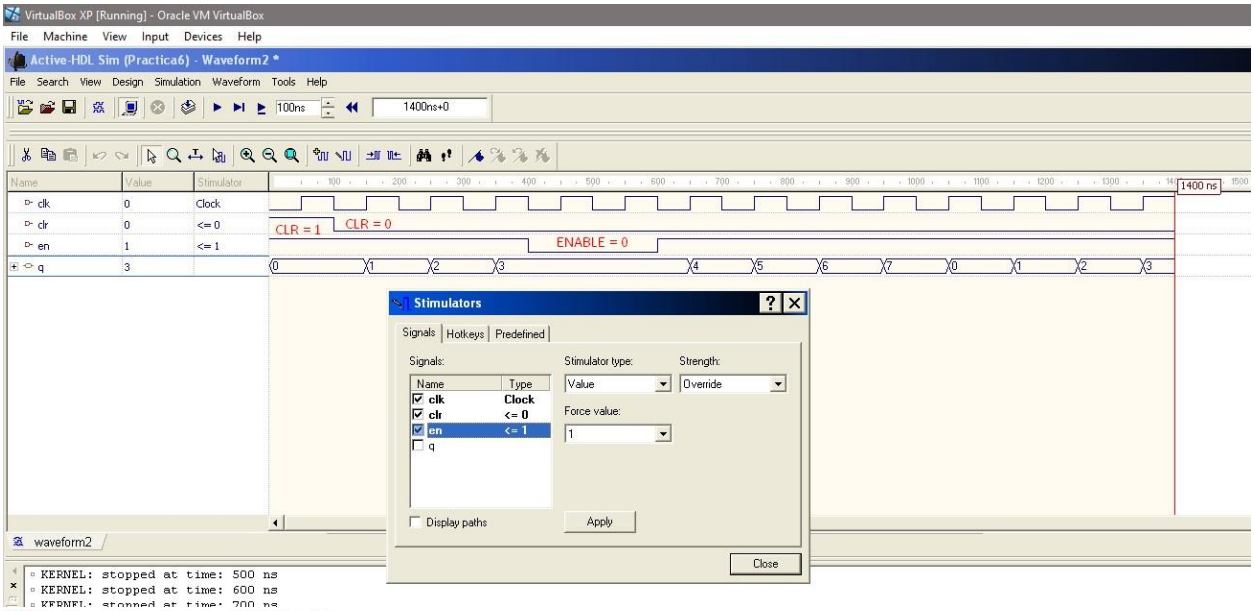
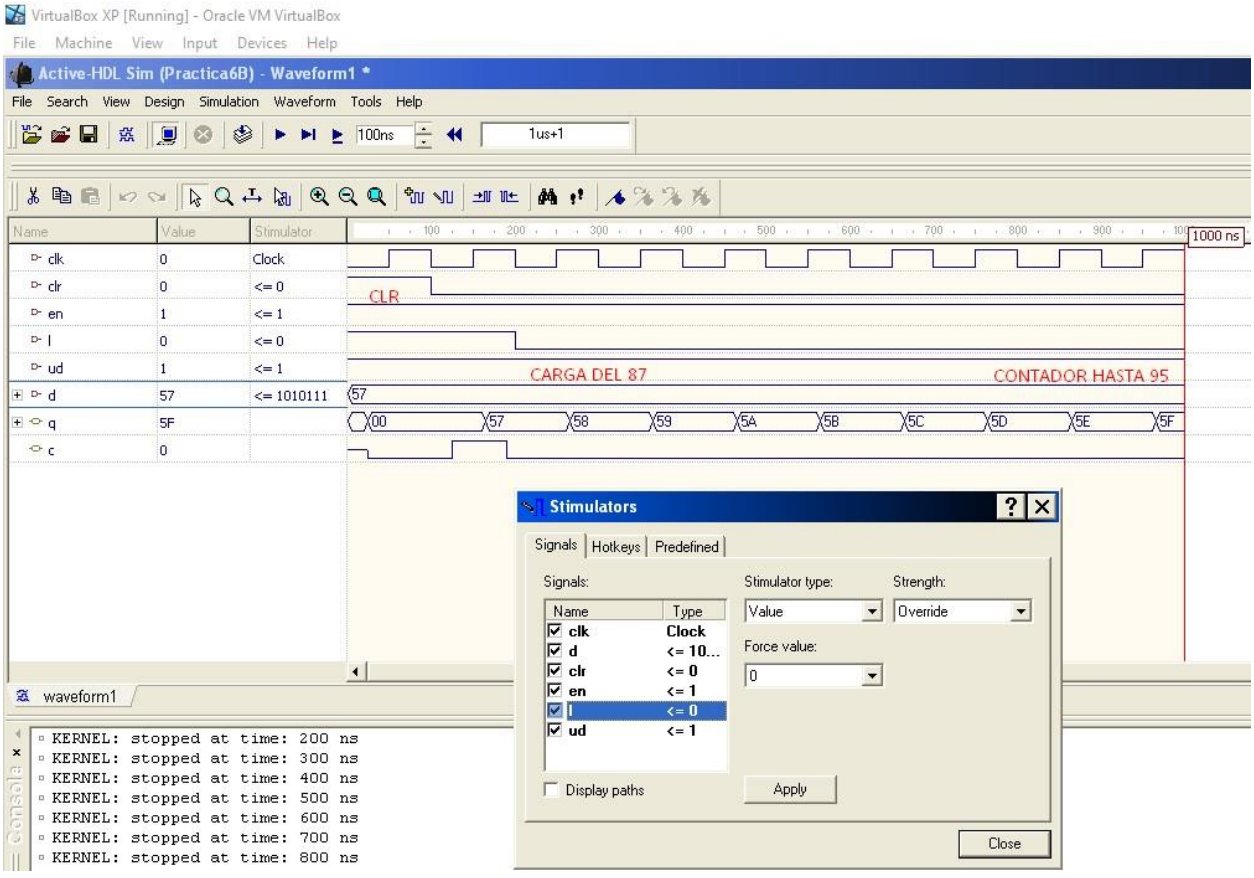
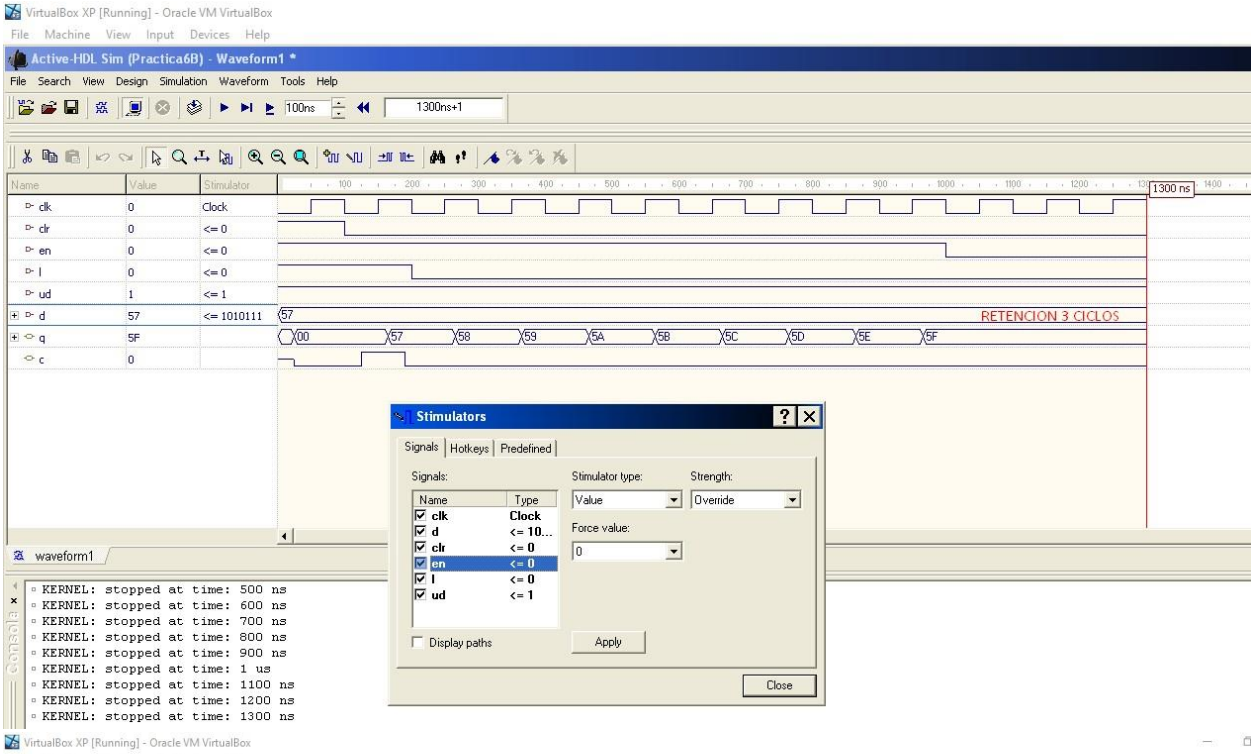
```
library ieee;
use ieee.std_logic_1164.all;

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;
```

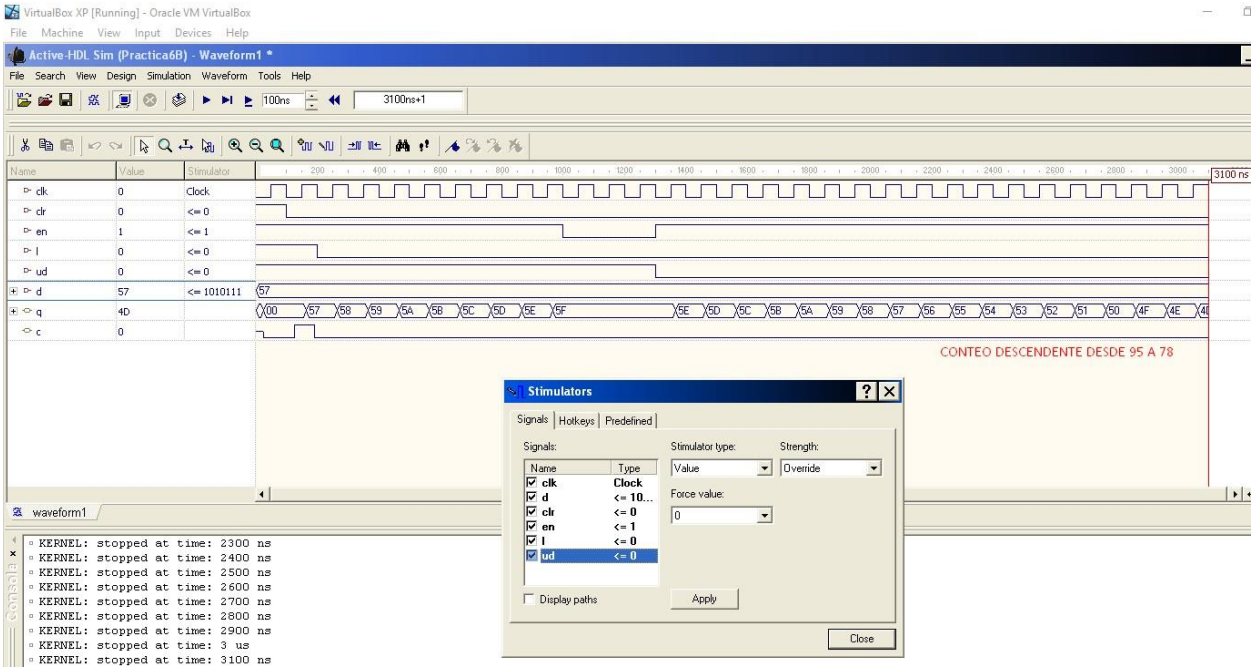
```
                                    end if;
                                    aux := aux and en; --toma todas la "operaciones and" que calculamos y le hace un and con ENABLE
                                    q(i) <= q(i) xor aux; -- terminamos la formula obtenida añadiendole el término en la 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
);
```

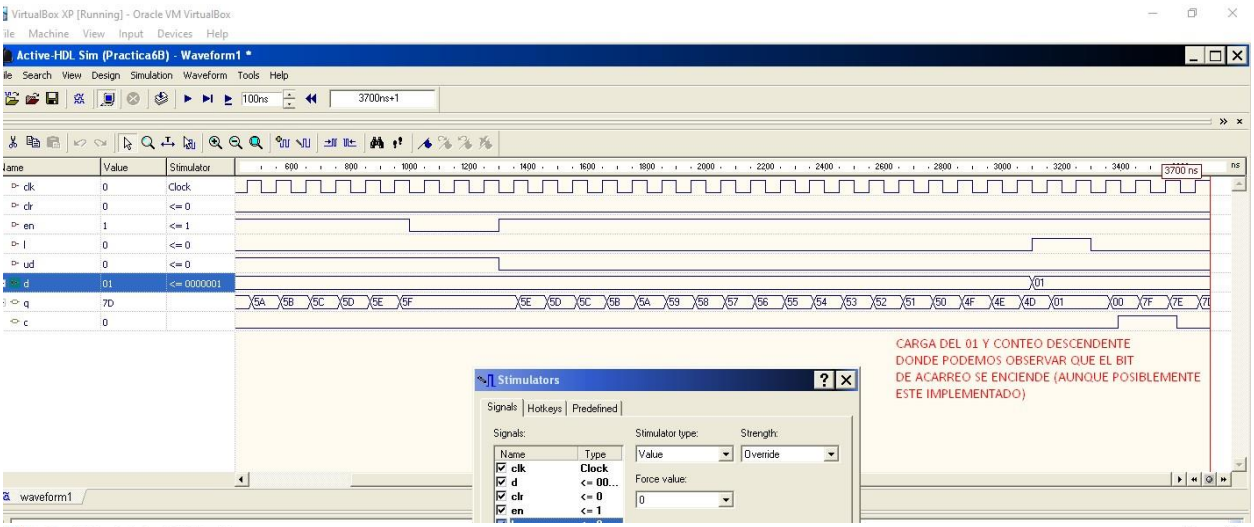
```
end practica6B;

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;
```

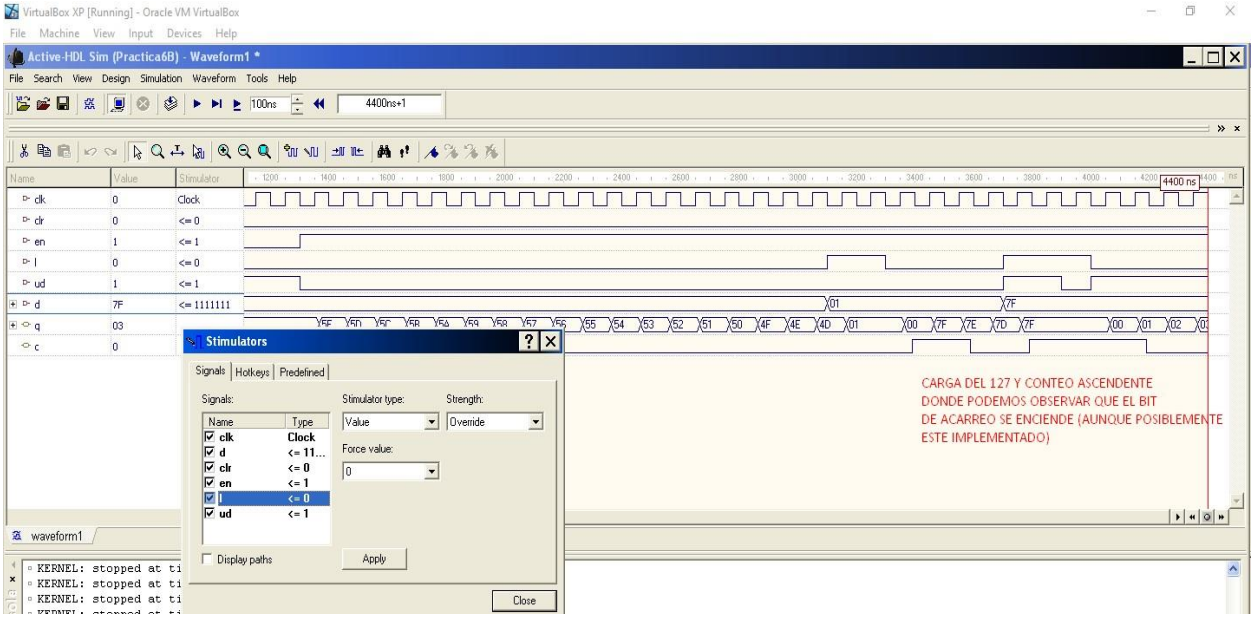

VirtualBox XP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Active-HDL Sim (Practica6) - Waveform2 *
File Search View Design Simulation Waveform Tools Help
100ns
1400ns+0
Name
Value
Stimulator
clk
clr
en
q
CLR = 1
CLR = 0
ENABLE = 0
1400 ns
Stimulators
Signals
Hotkeys
Predefined
Signals:
Stimulator type:
Strength:
Value
Override
Force value:
Display paths
Apply
Close
waveform2
KERNEL: stopped at time: 500 ns
KERNEL: stopped at time: 600 ns

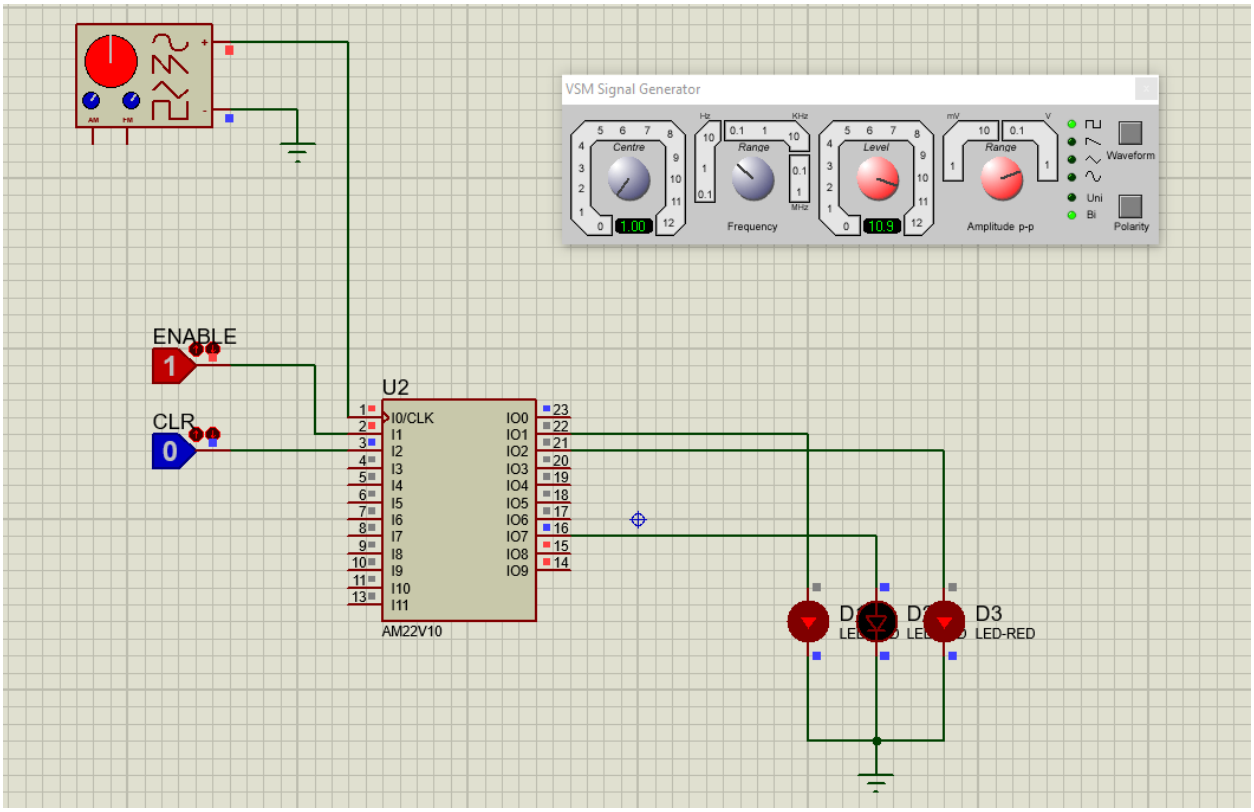

VirtualBox XP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Active-HDL Sim (Practica6B) - Waveform1 *
File Search View Design Simulation Waveform Tools Help
100ns
1us+1
Name
Value
Stimulator
clk
clr
en
l
ud
d
q
c
CLR
CARGA DEL 87
CONTADOR HASTA 95
1000 ns
Stimulators
Signals
Hotkeys
Predefined
Signals:
Stimulator type:
Strength:
Value
Override
Force value:
Display paths
Apply
Close
waveform1
KERNEL: stopped at time: 200 ns
KERNEL: stopped at time: 300 ns
KERNEL: stopped at time: 400 ns
KERNEL: stopped at time: 500 ns
KERNEL: stopped at time: 600 ns
KERNEL: stopped at time: 700 ns
KERNEL: stopped at time: 800 ns

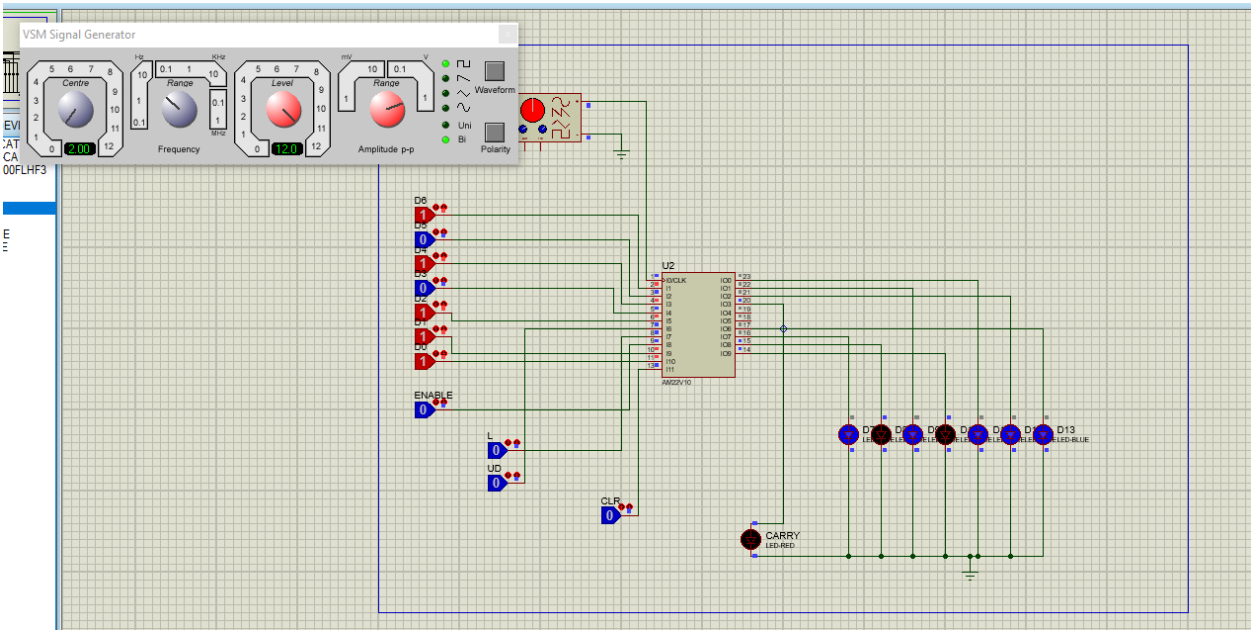

VirtualBox XP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Active-HDL Sim (Practica6B) - Waveform1 *
File Search View Design Simulation Waveform Tools Help
100ns
1300ns+1
Name
Value
Stimulator
clk 0 Clock
clr 0 <= 0
en 0 <= 0
l 0 <= 0
ud 1 <= 1
d 57 <= 1010111
q 5F
c 0
1300 ns
RETENCION 3 CICLOS
00 57 58 59 5A 5B 5C 5D 5E 5F
Stimulators
Signals Hotkeys Predefined
Signals:
Name Type
clk Clock
d <= 10...
clr <= 0
en <= 0
l <= 0
ud <= 1
Stimulator type:
Value
Strength:
Override
Force value:
0
Display paths
Apply
Close
waveform1
KERNEL: stopped at time: 500 ns
KERNEL: stopped at time: 600 ns
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
KERNEL: stopped at time: 1200 ns
KERNEL: stopped at time: 1300 ns

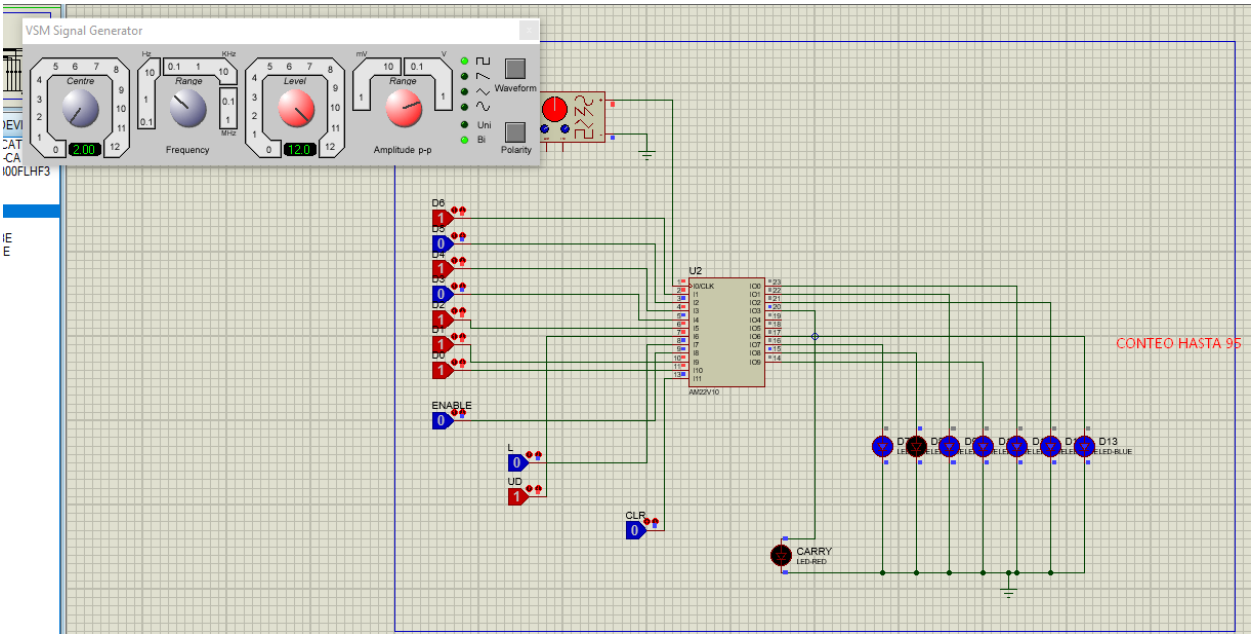

VirtualBox XP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Active-HDL Sim (Practica6B) - Waveform1 *
File Search View Design Simulation Waveform Tools Help
100ns
3100ns+1
Name
Value
Stimulator
clk 0 Clock
clr 0 <= 0
en 1 <= 1
l 0 <= 0
ud 0 <= 0
d 57 <= 1010111
q 4D
c 0
3100 ns
00 57 58 59 5A 5B 5C 5D 5E 5F 5E 5D 5C 5B 5A 59 58 57 56 55 54 53 52 51 50 4F 4E
CONTEO DESCENDENTE DESDE 95 A 78
Stimulators
Signals Hotkeys Predefined
Signals:
Name Type
clk Clock
d <= 10...
clr <= 0
en <= 1
l <= 0
ud <= 0
Stimulator type:
Value
Strength:
Override
Force value:
0
Display paths
Apply
Close
waveform1
KERNEL: stopped at time: 2300 ns
KERNEL: stopped at time: 2400 ns
KERNEL: stopped at time: 2500 ns
KERNEL: stopped at time: 2600 ns
KERNEL: stopped at time: 2700 ns
KERNEL: stopped at time: 2800 ns
KERNEL: stopped at time: 2900 ns
KERNEL: stopped at time: 3 us
KERNEL: stopped at time: 3100 ns

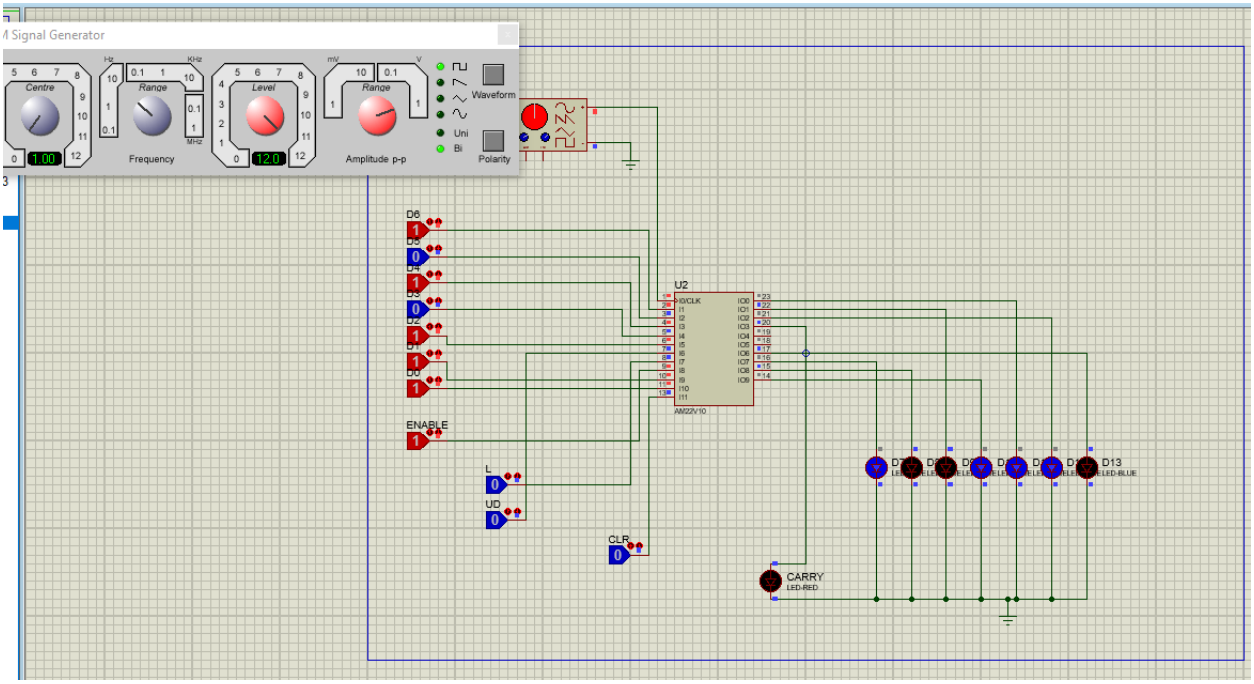

VirtualBox XP [Running] - Oracle VM VirtualBox
Active-HDL Sim (Practica6B) - Waveform1 *
Search View Design Simulation Waveform Tools Help
3700ns+1
Name
Value
Stimulator
clk 0 Clock
clr 0 <= 0
en 1 <= 1
l 0 <= 0
ud 0 <= 0
d 01 <= 0000001
q 7D
c 0
3700 ns
Stimulators
Signals
Hotkeys
Predefined
Signals:
Stimulator type:
Strength:
Value
Override
Force value:
0
CARGA DEL 01 Y CONTEO DESCENDENTE
DONDE PODEMOS OBSERVAR QUE EL BIT
DE ACARREO SE ENCIENDE (AUNQUE POSIBLEMENTE
ESTE IMPLEMENTADO)
waveform1

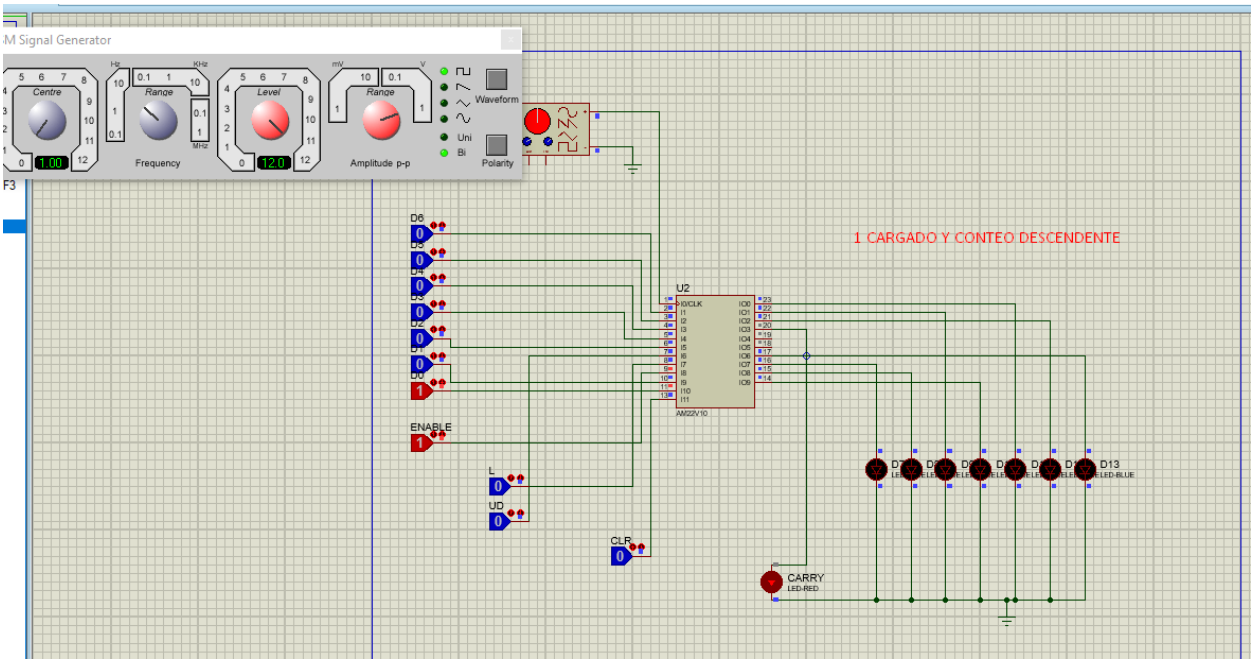

VirtualBox XP [Running] - Oracle VM VirtualBox
Active-HDL Sim (Practica6B) - Waveform1 *
File Search View Design Simulation Waveform Tools Help
4400ns+1
clk 0 Clock
clr 0 <= 0
en 1 <= 1
l 0 <= 0
ud 1 <= 1
d 7F <= 1111111
q 03
c 0
4400 ns
Stimulators
Signals
Hotkeys
Predefined
Signals:
Stimulator type:
Strength:
Value
Override
Force value:
0
clk Clock
d <= 11...
clr <= 0
en <= 1
l <= 0
ud <= 1
Display paths
Apply
Close
CARGA DEL 127 Y CONTEO ASCENDENTE
DONDE PODEMOS OBSERVAR QUE EL BIT
DE ACARREO SE ENCIENDE (AUNQUE POSIBLEMENTE
ESTE IMPLEMENTADO)
waveform1
KERNEL: stopped at ti
KERNEL: stopped at ti
KERNEL: stopped at ti

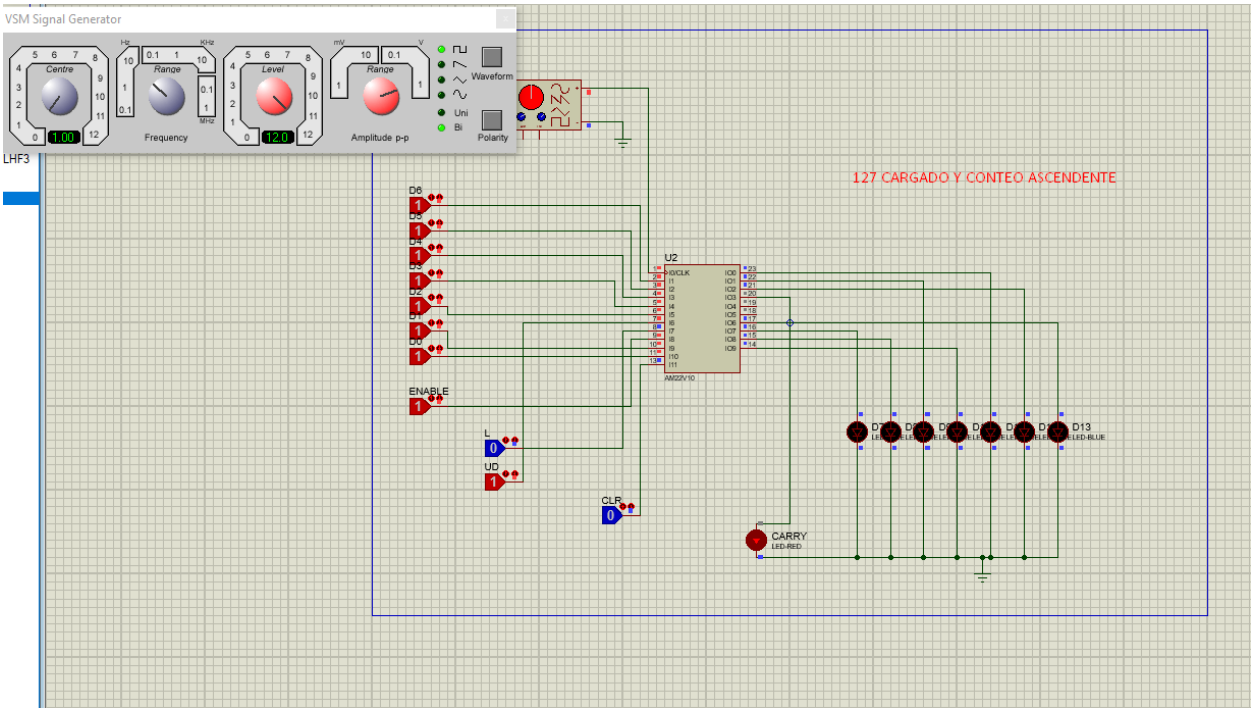

VSM Signal Generator
Centre
Range
Frequency
Level
Range
Amplitude p-p
Waveform
Uni
Bi
Polarity
ENABLE
CLR
U2
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
D3
LED-RED


VSM Signal Generator
Frequency
Amplitude p-p
U2
AM22V10
ENABLE
CLR
CARRY
LED-RED
LED-BLUE


VSM Signal Generator
Centre
Range
Frequency
Level
Range
Amplitude p-p
Waveform
Uni
Bi
Polarity
2.00
12.0
D6
ENABLE
L
UD
CLR
U2
AM22V10
CARRY
LED-RED
LED-BLUE
CONTEO HASTA 95


Signal Generator
Centre
Range
Frequency
Level
Range
Amplitude p-p
Waveform
Uni
Bi
Polarity
1.00
12.0
D6
ENABLE
L
UD
CLR
U2
AM22V10
CARRY
LED-RED
LED-BLUE


M Signal Generator
Centre
Range
Level
Range
Frequency
Amplitude p-p
Waveform
Uni
Bi
Polarity
1.00
12.0
F3
1 CARGADO Y CONTEO DESCENDENTE
D6
D5
D4
D3
D2
D1
D0
ENABLE
L
UD
CLR
U2
AM22V10
CARRY
LED-RED
D13
LED-BLUE


VSM Signal Generator
Centre
Range
Level
Range
Frequency
Amplitude p-p
Waveform
Uni
Bi
Polarity
1.00
12.0
LHF3
127 CARGADO Y CONTEO ASCENDENTE
D6
D5
D4
D3
D2
D1
D0
ENABLE
L
UD
CLR
U2
AM22V10
CARRY
LED-RED
D13
LED-BLUE

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.