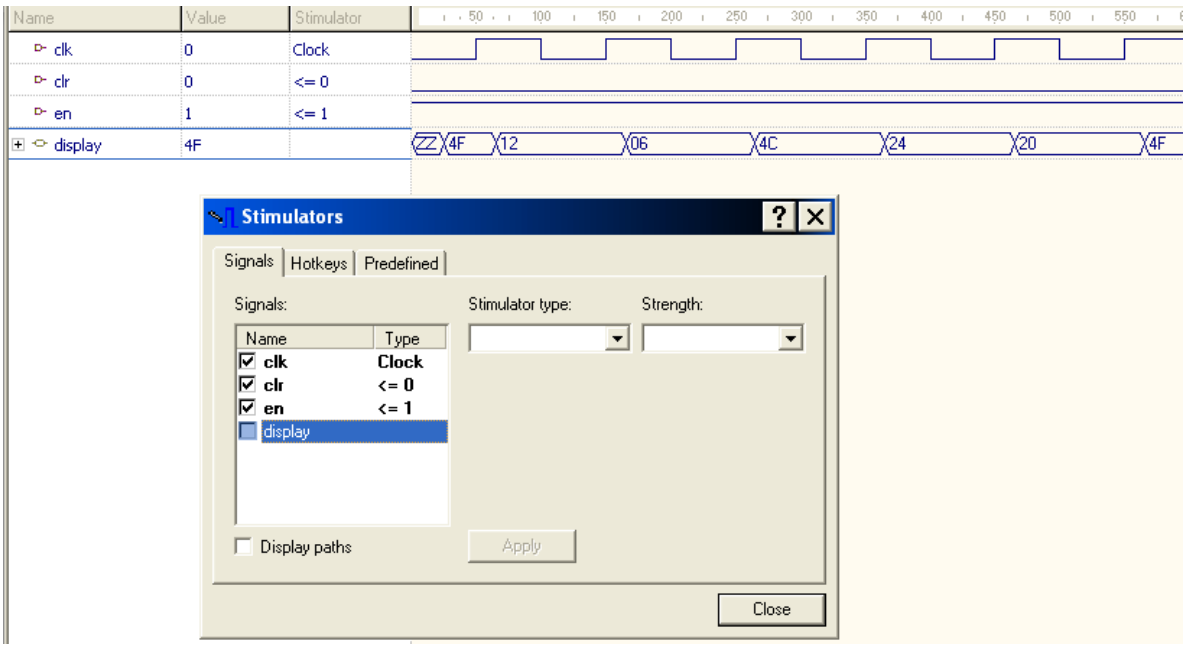
CIRCUITO 1

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

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

architecture aCont of contGen is -123456

constant edo0 : std_logic_vector( 6 downto 0) := "1001111";  --1
constant edo1 : std_logic_vector( 6 downto 0) := "0010010";  --2
constant edo2 : std_logic_vector( 6 downto 0) := "0000110";  --3
constant edo3 : std_logic_vector( 6 downto 0) := "1001100";  --4
constant edo4 : std_logic_vector( 6 downto 0) := "0100100";  --5
constant edo5 : std_logic_vector( 6 downto 0) := "0100000";  --6
signal estado : std_logic_vector(6 downto 0);

begin

    process(clk, clr)
    begin
        if (clr = '1') then
            estado <= edo0;
        elsif (rising_edge(clk)) then
            if (en = '1') then
                case estado is
                    when edo0 =>
                        estado <= edo1;
                    when edo1 =>
                        estado <= edo2;
                    when edo2 =>
                        estado <= edo3;
                    when edo3 =>
                        estado <= edo4;
                    when edo4 =>
                        estado <= edo5;
                    when others =>
                        estado <= edo0;
                end case;
            else
                estado <= estado;
            end if;
```

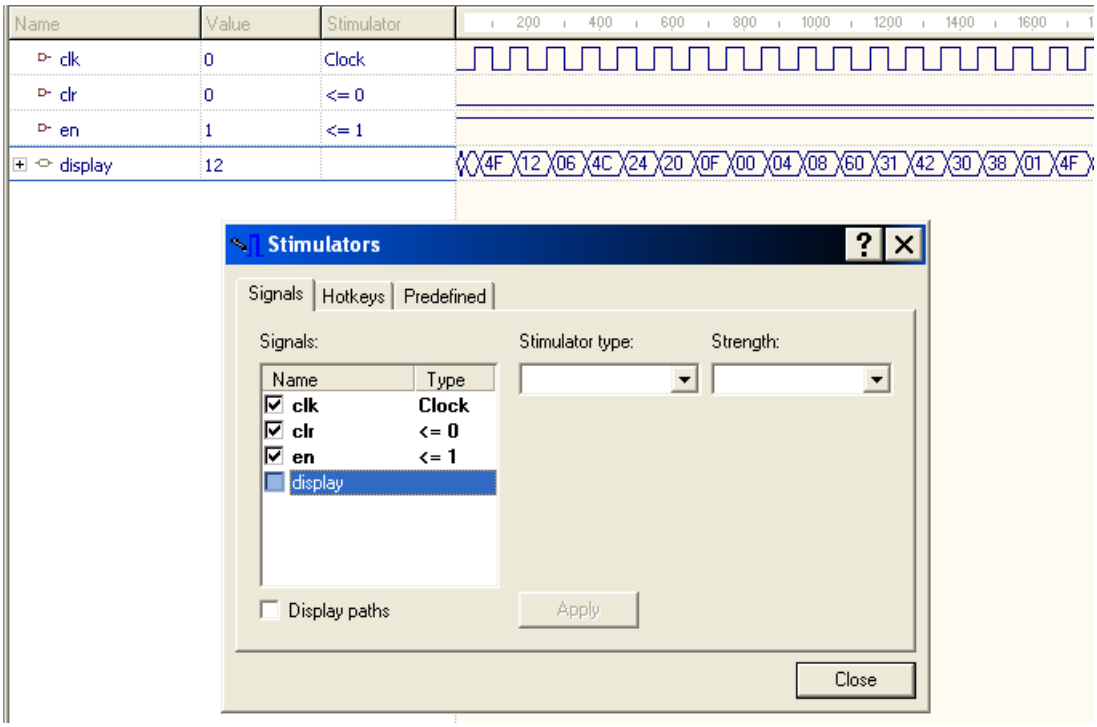
```
        end if;
    end process;

    display <= estado(6 downto 0);

end architecture;
```



CIRCUITO 2

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

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

architecture aCont of contGen is --0123456789AbCdEF

constant edo0 : std_logic_vector( 6 downto 0) := "0000001";--0
constant edo1 : std_logic_vector( 6 downto 0) := "1001111";--1
constant edo2 : std_logic_vector( 6 downto 0) := "0010010";--2
constant edo3 : std_logic_vector( 6 downto 0) := "0000110";--3
constant edo4 : std_logic_vector( 6 downto 0) := "1001100";--4
constant edo5 : std_logic_vector( 6 downto 0) := "0100100";--5
constant edo6 : std_logic_vector( 6 downto 0) := "0100000";--6
constant edo7 : std_logic_vector( 6 downto 0) := "0001111";--7
constant edo8 : std_logic_vector( 6 downto 0) := "0000000";--8
constant edo9 : std_logic_vector( 6 downto 0) := "0000100";--9
constant edo10 : std_logic_vector( 6 downto 0) := "0001000";--A
constant edo11 : std_logic_vector( 6 downto 0) := "1100000";--b
constant edo12 : std_logic_vector( 6 downto 0) := "0110001";--C
constant edo13 : std_logic_vector( 6 downto 0) := "1000010";--d
constant edo14 : std_logic_vector( 6 downto 0) := "0110000";--E
constant edo15 : std_logic_vector( 6 downto 0) := "0111000";--F

signal estado : std_logic_vector(6 downto 0);

begin

    process(clk, clr)
    begin
        if (clr = '1') then
            estado <= edo0;
        elsif (rising_edge(clk)) then
            if (en = '1') then
                case estado is
                    when edo0 =>
                        estado <= edo1;
                    when edo1 =>
                        estado <= edo2;
                    when edo2 =>
```

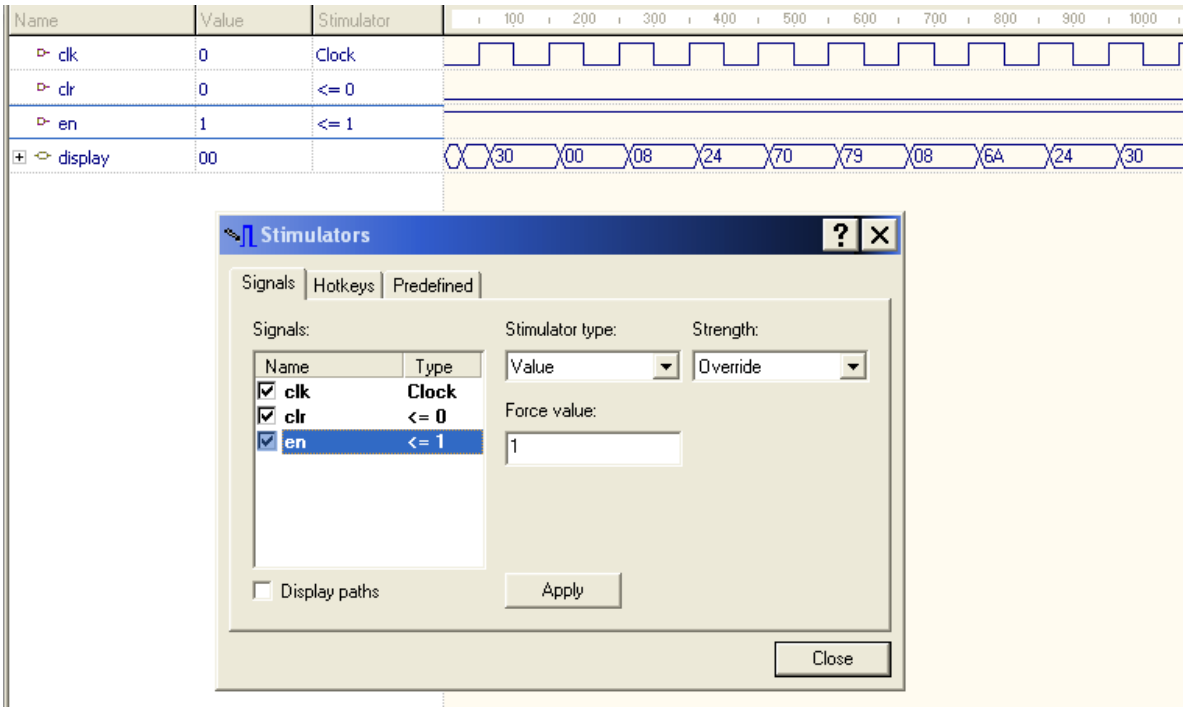
```
                        estado <= edo3;
                    when edo3 =>
                        estado <= edo4;
                    when edo4 =>
                        estado <= edo5;
                    when edo5 =>
                        estado <= edo6;
                    when edo6 =>
                        estado <= edo7;
                    when edo7 =>
                        estado <= edo8;
                    when edo8 =>
                        estado <= edo9;
                    when edo9 =>
                        estado <= edo10;
                    when edo10 =>
                        estado <= edo11;
                    when edo11 =>
                        estado <= edo12;
                    when edo12 =>
                        estado <= edo13;
                    when edo13 =>
                        estado <= edo14;
                    when edo14 =>
                        estado <= edo15;
                    when others =>
                        estado <= edo0;
                end case;
            else
                estado <= estado;
            end if;
        end if;
    end process;

    display <= estado(6 downto 0);

end architecture;
```


Name
Value
Stimulator
200
400
600
800
1000
1200
1400
1600
clk
0
Clock
clr
0
<= 0
en
1
<= 1
display
12
4F
12
06
4C
24
20
0F
00
04
08
60
31
42
30
38
01
4F
Stimulators
Signals
Hotkeys
Predefined
Signals:
Stimulator type:
Strength:
Name
Type
clk
Clock
clr
<= 0
en
<= 1
display
Display paths
Apply
Close

CIRCUITO 3

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

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

architecture aCont of contGen is --SEBAStIAn

constant s : std_logic_vector( 6 downto 0) := "0100100"; --0100100
constant e : std_logic_vector( 6 downto 0) := "0110000"; --0110000
constant b : std_logic_vector( 6 downto 0) := "0000000"; --0000000
constant a : std_logic_vector( 6 downto 0) := "0001000"; --0001000
constant t : std_logic_vector( 6 downto 0) := "1110000"; --1110000
constant i : std_logic_vector( 6 downto 0) := "1111001"; --1111001
constant n : std_logic_vector( 6 downto 0) := "1101010"; --1101010

constant e0 : std_logic_vector(1 downto 0) :="00";
constant e1 : std_logic_vector(1 downto 0) :="01";
constant e2 : std_logic_vector(1 downto 0) :="10";

constant edo0 : std_logic_vector( 8 downto 0) := e0 & s;  --S
constant edo1 : std_logic_vector( 8 downto 0) := e0 & e;  --E
constant edo2 : std_logic_vector( 8 downto 0) := e0 & b;  --B
constant edo3 : std_logic_vector( 8 downto 0) := e1 & a;  --A
constant edo4 : std_logic_vector( 8 downto 0) := e1 & s;  --S
constant edo5 : std_logic_vector( 8 downto 0) := e0 & t;  --t
constant edo6 : std_logic_vector( 8 downto 0) := e0 & i;  --I
constant edo7 : std_logic_vector( 8 downto 0) := e2 & a;  --A
constant edo8 : std_logic_vector( 8 downto 0) := e0 & n;  --n

signal estado : std_logic_vector(8 downto 0);

begin

    process(clk, clr)
    begin
        if (clr = '1') then
            estado <= edo0;
        elsif (rising_edge(clk)) then
            if (en = '1') then
                case estado is
```

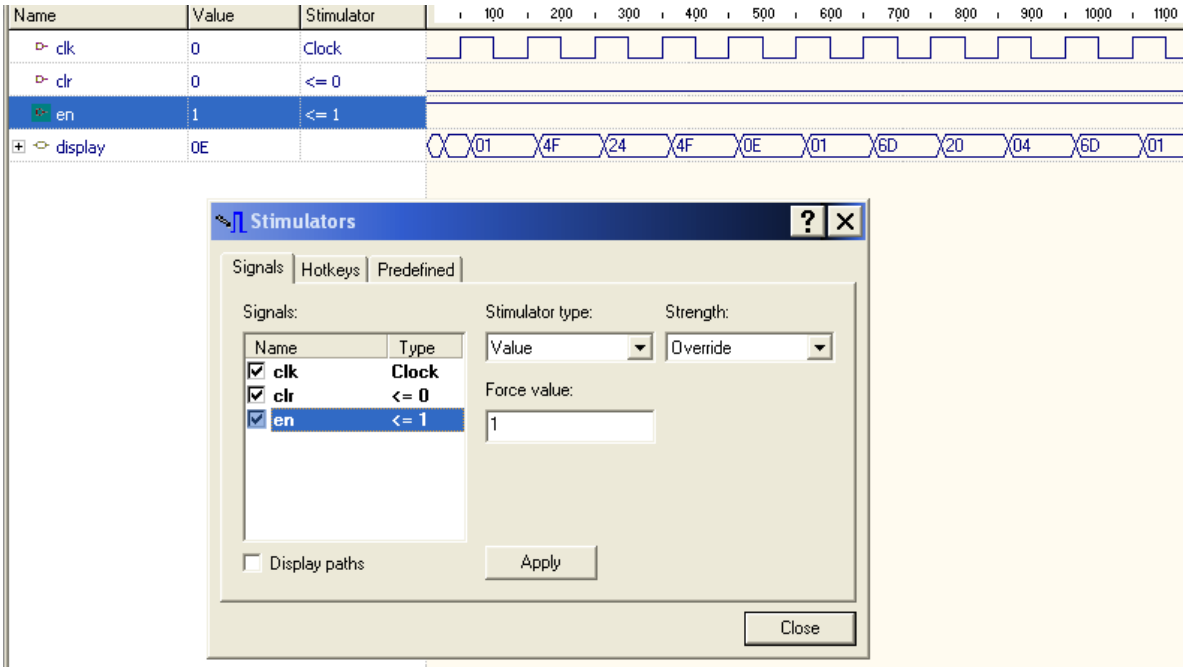
```
                    when edo0 =>
                        estado <= edo1;
                    when edo1 =>
                        estado <= edo2;
                    when edo2 =>
                        estado <= edo3;
                    when edo3 =>
                        estado <= edo4;
                    when edo4 =>
                        estado <= edo5;
                    when edo5 =>
                        estado <= edo6;
                    when edo6 =>
                        estado <= edo7;
                    when edo7 =>
                        estado <= edo8;
                    when others =>
                        estado <= edo0;
                end case;
            else
                estado <= estado;
            end if;
        end if;
    end process;

    display <= estado(6 downto 0);

end architecture;
```


Name
Value
Stimulator
100
200
300
400
500
600
700
800
900
1000
clk
0
Clock
clr
0
<= 0
en
1
<= 1
display
00
30
00
08
24
70
79
08
6A
24
30
Stimulators
Signals
Hotkeys
Predefined
Signals:
Name
Type
clk
Clock
clr
<= 0
en
<= 1
Stimulator type:
Value
Strength:
Override
Force value:
1
Display paths
Apply
Close

CODIGO 4

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

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

architecture aCont of contGen is --2015170269

constant b2 : std_logic_vector( 6 downto 0) := "1101101"; --0010010
constant b0 : std_logic_vector( 6 downto 0) := "0000001"; --0000001
constant b1 : std_logic_vector( 6 downto 0) := "1001111"; --1001111
constant b7 : std_logic_vector( 6 downto 0) := "0001110"; --0001110
constant b5 : std_logic_vector( 6 downto 0) := "0100100"; --0100100
constant b6 : std_logic_vector( 6 downto 0) := "0100000"; --0100000
constant b9 : std_logic_vector( 6 downto 0) := "0000100"; --0000100

constant e0 : std_logic :='0';
constant e1 : std_logic :='1';

constant edo0 : std_logic_vector( 7 downto 0) := e0 & b2;  --2
constant edo1 : std_logic_vector( 7 downto 0) := e0 & b0;  --0
constant edo2 : std_logic_vector( 7 downto 0) := e0 & b1;  --1
constant edo3 : std_logic_vector( 7 downto 0) := e0 & b5;  --5
constant edo4 : std_logic_vector( 7 downto 0) := e1 & b1;  --1
constant edo5 : std_logic_vector( 7 downto 0) := e0 & b7;  --7
constant edo6 : std_logic_vector( 7 downto 0) := e1 & b0;  --0
constant edo7 : std_logic_vector( 7 downto 0) := e1 & b2;  --2
constant edo8 : std_logic_vector( 7 downto 0) := e0 & b6;  --6
constant edo9 : std_logic_vector( 7 downto 0) := e0 & b9;  --9

signal estado : std_logic_vector(7 downto 0);

begin

    process(clk, clr)
    begin
        if (clr = '1') then
            estado <= edo0;
        elsif (rising_edge(clk)) then
            if (en = '1') then
                case estado is
```

```
                    when edo0 =>
                        estado <= edo1;
                    when edo1 =>
                        estado <= edo2;
                    when edo2 =>
                        estado <= edo3;
                    when edo3 =>
                        estado <= edo4;
                    when edo4 =>
                        estado <= edo5;
                    when edo5 =>
                        estado <= edo6;
                    when edo6 =>
                        estado <= edo7;
                    when edo7 =>
                        estado <= edo8;
                    when edo8 =>
                        estado <= edo9;
                    when others =>
                        estado <= edo0;
                end case;
            else
                estado <= estado;
            end if;
        end if;
    end process;

    display <= estado(6 downto 0);

end architecture;
```


Name
Value
Stimulator
100
200
300
400
500
600
700
800
900
1000
1100
clk
0
Clock
clr
0
<= 0
en
1
<= 1
display
0E
01
4F
24
4F
0E
01
6D
20
04
6D
01
Stimulators
Signals
Hotkeys
Predefined
Signals:
Name
Type
clk
Clock
clr
<= 0
en
<= 1
Stimulator type:
Value
Strength:
Override
Force value:
1
Display paths
Apply
Close

CUESTIONARIO

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

R= 4

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

R= 10-20 por cada circuito

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

R= 3 de entrada y 7 de salida en todos

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?

R= Para el primero: 10 / 22 = 45 %

Para el segundo: 42 / 121 = 34 %

Para el tercero: 31 / 121 = 25 %

Para el cuarto: 32 / 121 = 26 %

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

R= No, en la mayoría fue necesario utilizar la definida por el usuario

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

R= El reloj de manera síncrona y las demás asíncronas

7. ¿Qué puedes concluir de esta práctica?

R= Para poder decodificar un mensaje es necesario que no existan valores repetidos, de otra manera el autómata diseñado se confundiría de estado y regresaría el que ya existe. La mejor manera para representarlo es con base a un sistema definido por el usuario, ya que algunos que existen necesitan más salidas de las que nuestro circuito puede soportar.