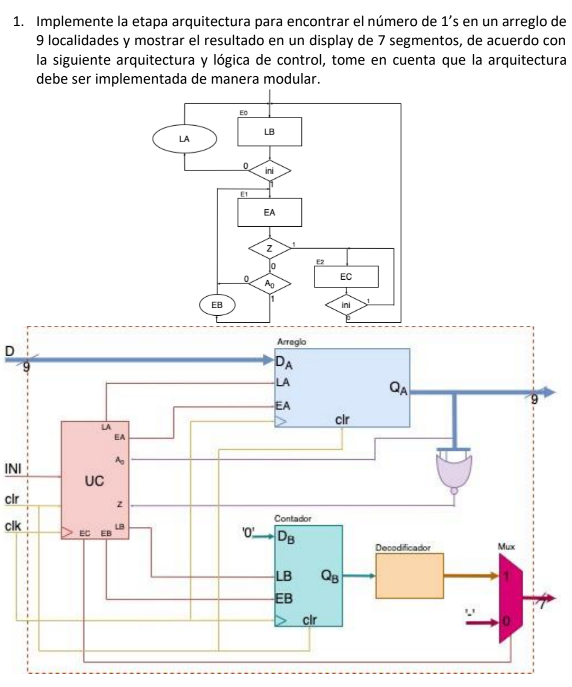
# Práctica 12

# Cartas ASM





## Unidad de Control

### Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity ctrlUnit is

    Port(

    clr, clk: in std\_logic;

    ini, z, a0: in std\_logic;

    la, lb, ea,eb,ec : out std\_logic);

end ctrlUnit;

architecture Behavioral of ctrlUnit is

type estados is(E0, E1, E2);

signal edo\_act, edo\_sig: estados;

begin

    process(clk, clr)

    begin

    if(clr = '1') then

        edo\_act <= E0;

    elsif(rising\_edge(clk)) then

    edo\_act <= edo\_sig;

    end if;

    end process;

    process(edo\_act, ini,z, a0)

    begin

        la <= '0';

        lb <= '0';

        ea <= '0';

        eb <= '0';

        ec <= '0';

        case edo\_act is

            when E0 =>

            lb <= '1';

            if(ini = '0') then

                la <= '1';

                edo\_sig <= E0;

            else

                edo\_sig <= E1;

            end if;

            when E1 =>

                ea <= '1';

                if (z = '0') then

                    if (a0 = '0') then

                        edo\_sig <= E1;

                    else

                        eb <= '1';

                        edo\_sig <= E1;

                    end if;

                else

                    edo\_sig <= E2;

                end if;

            when E2 =>

                ec <= '1';

                if (ini = '0') then

                    edo\_sig <= E0;

                else

                    edo\_sig <= E2;

                end if;

        end case;

    end process;

end Behavioral;

### Código de simulación y forma de onda

library ieee;

use ieee.std\_logic\_1164.all;

entity tb\_ctrlUnit is

end tb\_ctrlUnit;

architecture tb of tb\_ctrlUnit is

    component ctrlUnit

        port (clr : in std\_logic;

              clk : in std\_logic;

              ini : in std\_logic;

              z   : in std\_logic;

              a0  : in std\_logic;

              la  : out std\_logic;

              lb  : out std\_logic;

              ea  : out std\_logic;

              eb  : out std\_logic;

              ec  : out std\_logic);

    end component;

    signal clr : std\_logic;

    signal clk : std\_logic;

    signal ini : std\_logic;

    signal z   : std\_logic;

    signal a0  : std\_logic;

    signal la  : std\_logic;

    signal lb  : std\_logic;

    signal ea  : std\_logic;

    signal eb  : std\_logic;

    signal ec  : std\_logic;

    constant TbPeriod : time := 1000 ns; *-- EDIT Put right period here*

    signal TbClock : std\_logic := '0';

    signal TbSimEnded : std\_logic := '0';

begin

    dut : ctrlUnit

    port map (clr => clr,

              clk => clk,

              ini => ini,

              z   => z,

              a0  => a0,

              la  => la,

              lb  => lb,

              ea  => ea,

              eb  => eb,

              ec  => ec);

*-- Clock generation*

    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

*-- EDIT: Check that clk is really your main clock signal*

    clk <= TbClock;

    stimuli : process

    begin

*-- EDIT Adapt initialization as needed*

        clr <= '0';

        ini <= '1';

        z <= '0';

        a0 <= '1';

*-- EDIT Add stimuli here*

        wait for 100 \* TbPeriod;

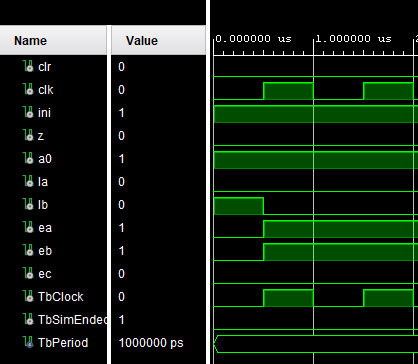
*-- Stop the clock and hence terminate the simulation*

        TbSimEnded <= '1';

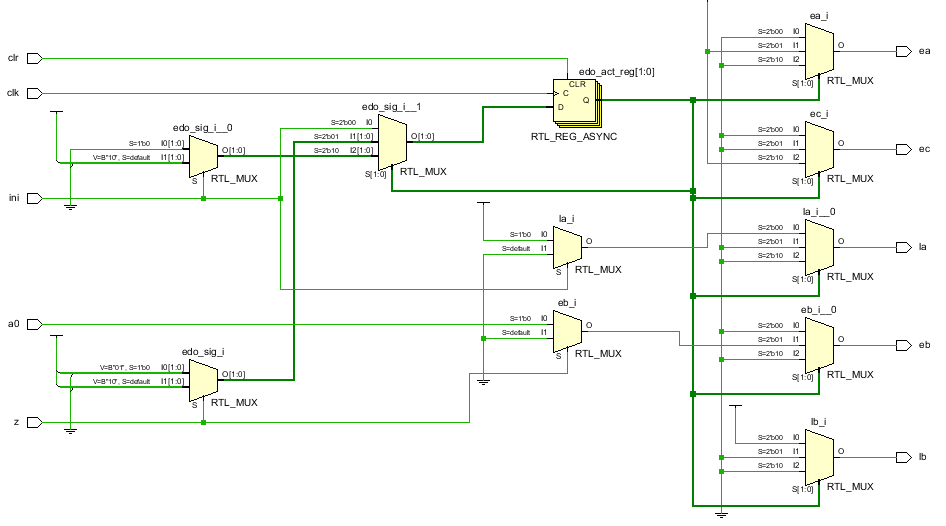
        wait;

    end process;

end tb;



### Diagrama RTL



## Registro

### Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity arreglo is

Port ( la, ea, clk, clr : in  STD\_LOGIC;

        dato : in  STD\_LOGIC\_VECTOR (8 downto 0);

        dato1 : out  STD\_LOGIC\_VECTOR (8 downto 0);

        z, a0 : out  STD\_LOGIC);

end arreglo;

architecture Behavioral of arreglo is

signal a: STD\_LOGIC\_VECTOR (8 downto 0);

begin

    process (clk, clr)

        begin

            if (clr = '1') then

                a<="000000000";

            elsif (clk'event and clk = '1') then

                if (la = '1' and ea = '0') then

                    a <= dato;

                end if;

                if (la = '0' and ea = '1') then

                    a <= to\_stdlogicvector(to\_bitvector(a) srl 1);

                end if;

            end if;

    end process;

    dato1 <= a;

    a0 <= a(0);

    z <= not(a(0) or a(1) or a(2) or a(3) or a(4) or a(5) or a(6) or a(7) or a(8));

end Behavioral;

### Código de simulación y forma de onda

library ieee;

use ieee.std\_logic\_1164.all;

entity tb\_arreglo is

end tb\_arreglo;

architecture tb of tb\_arreglo is

    component arreglo

        port (la    : in std\_logic;

              ea    : in std\_logic;

              clk   : in std\_logic;

              clr   : in std\_logic;

              dato  : in std\_logic\_vector (8 downto 0);

              salida : out std\_logic\_vector (8 downto 0);

              z     : out std\_logic;

              a0    : out std\_logic);

    end component;

    signal la    : std\_logic;

    signal ea    : std\_logic;

    signal clk   : std\_logic;

    signal clr   : std\_logic;

    signal dato  : std\_logic\_vector (8 downto 0);

    signal salida : std\_logic\_vector (8 downto 0);

    signal z     : std\_logic;

    signal a0    : std\_logic;

    constant TbPeriod : time := 10 ns; *-- EDIT Put right period here*

    signal TbClock : std\_logic := '0';

    signal TbSimEnded : std\_logic := '0';

begin

    dut : arreglo

    port map (la    => la,

              ea    => ea,

              clk   => clk,

              clr   => clr,

              dato  => dato,

              salida => salida,

              z     => z,

              a0    => a0);

TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

*-- EDIT: Check that clk is really your main clock signal*

    clk <= TbClock;

    stimuli : process

    begin

            wait for 10 ns;

            dato<= "111111111";

            la <= '1';

            ea <= '0';

            clr<= '0';

            wait for 10 ns;

            clr <= '0';

            la<= '0';

            ea <= '1';

            wait for 10ns;

            la <= '0';

            ea <= '1';

            wait for 10ns;

            la <= '0';

            ea <= '1';

            wait for 10ns;

            la <= '0';

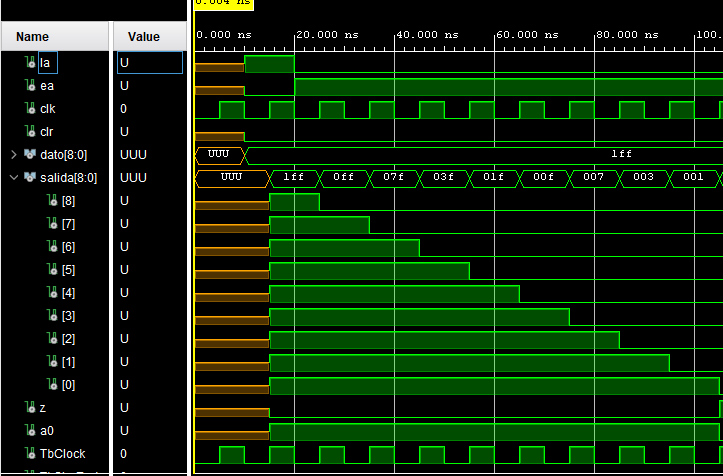
            ea <= '1';

            wait for 10ns;

        wait;

    end process;

end tb;



### Diagrama RTL

## 

## Contador

### Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Contador is

Port ( clk, clr : in  STD\_LOGIC;

       lb, eb : in  STD\_LOGIC;

       B : inout  STD\_LOGIC\_VECTOR (3 downto 0));

end Contador;

architecture Behavioral of Contador is

begin

    process(clk,clr)

        begin

            if(clr = '1' ) then

                B <= "0000";

            elsif(clk'event and clk = '1' ) then

                if(eb = '0' and lb = '1') then

                    B <= "0000";

                elsif(eb = '1' and lb = '0' ) then

                    B <= B + 1;

                end if;

            end if;

    end process;

end Behavioral;

### Código de simulación y forma de onda

library ieee;

use ieee.std\_logic\_1164.all;

entity tb\_Contador is

end tb\_Contador;

architecture tb of tb\_Contador is

    component Contador

        port (clk : in std\_logic;

              clr : in std\_logic;

              lb  : in std\_logic;

              eb  : in std\_logic;

              B   : inout std\_logic\_vector (3 downto 0));

    end component;

    signal clk : std\_logic;

    signal clr : std\_logic;

    signal lb  : std\_logic;

    signal eb  : std\_logic;

    signal B   : std\_logic\_vector (3 downto 0);

    constant TbPeriod : time := 100 ns; *-- EDIT Put right period here*

    signal TbClock : std\_logic := '0';

    signal TbSimEnded : std\_logic := '0';

begin

    dut : Contador

    port map (clk => clk,

              clr => clr,

              lb  => lb,

              eb  => eb,

              B   => B);

*-- Clock generation*

 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

*-- EDIT: Check that clk is really your main clock signal*

    clk <= TbClock;

    stimuli : process

    begin

*-- EDIT Adapt initialization as needed*

        wait for 10ns;

        clr <= '0';

        wait for 100ns;

        lb <= '1';

        eb <= '0';

        wait for 100ns;

        lb <= '0';

        eb <= '1';

        wait for 100ns;

        lb <= '0';

        eb <= '0';

        wait for 100ns;

        lb <= '0';

        eb <= '1';

        wait for 200ns;

        lb <= '0';

        eb <= '0';

        wait for 100ns;

*-- EDIT Add stimuli here*

        wait for 100 \* TbPeriod;

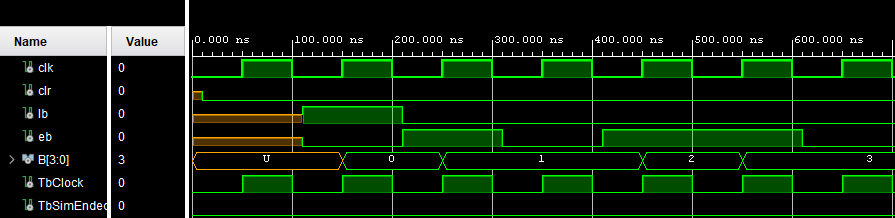
*-- Stop the clock and hence terminate the simulation*

        TbSimEnded <= '1';

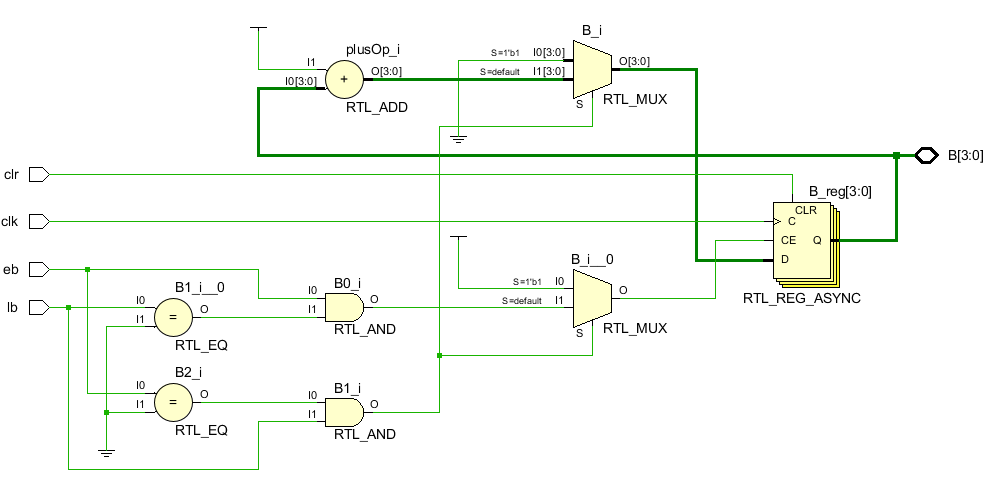
        wait;

    end process;

end tb;



### Diagrama RTL



## Multiplexor

### Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity multiplexor is

 Port ( aux : in  STD\_LOGIC\_VECTOR (6 downto 0);

           ec : in  STD\_LOGIC;

           display : out  STD\_LOGIC\_VECTOR (6 downto 0));

end multiplexor;

architecture Behavioral of multiplexor is

constant guion : std\_logic\_vector(6 downto 0) := "1111110";

begin

    with ec select

        display <= guion when '0',

                    aux when others;

end Behavioral;

### Código de simulación y forma de onda

library ieee;

use ieee.std\_logic\_1164.all;

entity tb\_multiplexor is

end tb\_multiplexor;

architecture tb of tb\_multiplexor is

    component multiplexor

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

              ec      : in std\_logic;

              display : out std\_logic\_vector (6 downto 0));

    end component;

    signal aux     : std\_logic\_vector (6 downto 0);

    signal ec      : std\_logic;

    signal display : std\_logic\_vector (6 downto 0);

    constant TbPeriod : time := 10 ns; *-- EDIT Put right period here*

    signal TbClock : std\_logic := '0';

    signal TbSimEnded : std\_logic := '0';

begin

    dut : multiplexor

    port map (aux     => aux,

              ec      => ec,

              display => display);

*-- Clock generation*

TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

    stimuli : process

    begin

        wait for 10ns;

        aux <= "1001111";

        ec <= '0';

        wait for 10ns;

        ec <= '1';

        wait for 10ns;

*-- EDIT Add stimuli here*

        wait for 100 \* TbPeriod;

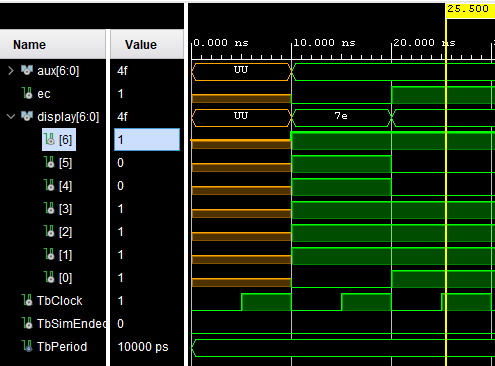
*-- Stop the clock and hence terminate the simulation*

        TbSimEnded <= '1';

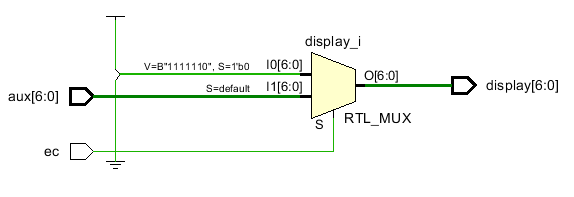
        wait;

    end process;

end tb;



### Diagrama RTL



## Decodificador

### Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Decodificador is

Port ( B : in  STD\_LOGIC\_VECTOR (3 downto 0);

           aux : out  STD\_LOGIC\_VECTOR (6 downto 0));

end Decodificador;

architecture Behavioral of Decodificador is

    constant num\_0 : std\_logic\_vector(6 downto 0) := "0000001";

    constant num\_1 : std\_logic\_vector(6 downto 0) := "1001111";

    constant num\_2 : std\_logic\_vector(6 downto 0) := "0010010";

    constant num\_3 : std\_logic\_vector(6 downto 0) := "0000110";

    constant num\_4 : std\_logic\_vector(6 downto 0) := "1001100";

    constant num\_5 : std\_logic\_vector(6 downto 0) := "0100100";

    constant num\_6 : std\_logic\_vector(6 downto 0) := "0100000";

    constant num\_7 : std\_logic\_vector(6 downto 0) := "0001111";

    constant num\_8 : std\_logic\_vector(6 downto 0) := "0000000";

    constant num\_9 : std\_logic\_vector(6 downto 0) := "0000100";

    constant guion : std\_logic\_vector(6 downto 0) := "1111110";

begin

    with B select

        aux <= num\_0 when "0000",

                num\_1 when "0001",

                num\_2 when "0010",

                num\_3 when "0011",

                num\_4 when "0100",

                num\_5 when "0101",

                num\_6 when "0110",

                num\_7 when "0111",

                num\_8 when "1000",

                num\_9 when "1001",

                guion when others;

end Behavioral;

### Código de simulación y forma de onda

library ieee;

use ieee.std\_logic\_1164.all;

entity tb\_Decodificador is

end tb\_Decodificador;

architecture tb of tb\_Decodificador is

    component Decodificador

        port (B   : in std\_logic\_vector (3 downto 0);

              aux : out std\_logic\_vector (6 downto 0));

    end component;

    signal B   : std\_logic\_vector (3 downto 0);

    signal aux : std\_logic\_vector (6 downto 0);

    constant TbPeriod : time := 10 ns; *-- EDIT Put right period here*

    signal TbClock : std\_logic := '0';

    signal TbSimEnded : std\_logic := '0';

begin

    dut : Decodificador

    port map (B   => B,

              aux => aux);

*-- Clock generation*

TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

    stimuli : process

    begin

*-- EDIT Adapt initialization as needed*

        wait for 10ns;

        B <= "0010";

        wait for 10ns;

*-- EDIT Add stimuli here*

        wait for 100 \* TbPeriod;

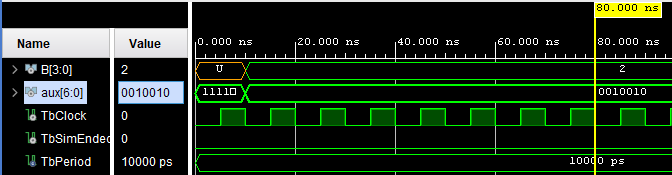
*-- Stop the clock and hence terminate the simulation*

        TbSimEnded <= '1';

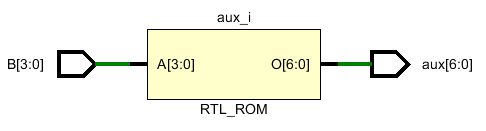
        wait;

    end process;

end tb;



### Diagrama RTL



## CartaASM

### Código de implementación

library IEEE;

library work;

use IEEE.STD\_LOGIC\_1164.ALL;

use work.mypackage.ALL;

entity cartaASM is

Port ( clr, clk : in  STD\_LOGIC;

           ini : in  STD\_LOGIC;

           dato : in  STD\_LOGIC\_VECTOR (8 downto 0);

           dato1 : out  STD\_LOGIC\_VECTOR (8 downto 0);

           display : out  STD\_LOGIC\_VECTOR (6 downto 0));

end cartaASM;

architecture Behavioral of cartaASM is

signal lb, eb, la, ea, ec, a0, z : STD\_LOGIC;

signal B : STD\_LOGIC\_VECTOR (3 downto 0);

signal aux : STD\_LOGIC\_VECTOR (6 downto 0);

begin

    unic : ctrlUnit port map (

        clk => clk,

        clr => clr,

        ini => ini,

        a0 => a0,

        z => z,

        la => la,

        ea => ea,

        eb => eb,

        lb => lb,

        ec => ec

    );

    reg : arreglo port map (

        clk => clk,

        clr => clr,

        la => la,

        ea => ea,

        dato => dato,

        dato1 => dato1,

        z => z,

        a0 => a0

    );

    conta : Contador port map (

        clk => clk,

        clr => clr,

        lb => lb,

        eb => eb,

        B => B

    );

    deco : Decodificador port map (

        B => B,

        aux => aux

    );

    muxe : multiplexor port map (

        aux => aux,

        ec => ec,

        display => display

    );

end Behavioral;

### Código de simulación

library ieee;

use ieee.std\_logic\_1164.all;

entity tb\_cartaASM is

end tb\_cartaASM;

architecture tb of tb\_cartaASM is

    component cartaASM

        port (clr     : in std\_logic;

              clk     : in std\_logic;

              ini     : in std\_logic;

              dato    : in std\_logic\_vector (8 downto 0);

              dato1   : out std\_logic\_vector (8 downto 0);

              display : out std\_logic\_vector (6 downto 0));

    end component;

    signal clr     : std\_logic;

    signal clk     : std\_logic;

    signal ini     : std\_logic;

    signal dato    : std\_logic\_vector (8 downto 0);

    signal dato1   : std\_logic\_vector (8 downto 0);

    signal display : std\_logic\_vector (6 downto 0);

    constant TbPeriod : time := 100 ns; *-- EDIT Put right period here*

    signal TbClock : std\_logic := '0';

    signal TbSimEnded : std\_logic := '0';

begin

    dut : cartaASM

    port map (clr     => clr,

              clk     => clk,

              ini     => ini,

              dato    => dato,

              dato1   => dato1,

              display => display);

*-- Clock generation*

    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

*-- EDIT: Check that clk is really your main clock signal*

    clk <= TbClock;

    stimuli : process

    begin

        ini <= '0';

*--dato <= "101101011";    --a*

*--dato <= "000011101";    --b*

*--dato <= "000010000";    --c*

*--dato <= "100001000";    --d*

        dato <= "000000000";    *--e*

        wait for 100ns;

        clr <= '0';

        ini <='1';

*-- EDIT Add stimuli here*

        wait for 12 \* TbPeriod;

*-- Stop the clock and hence terminate the simulation*

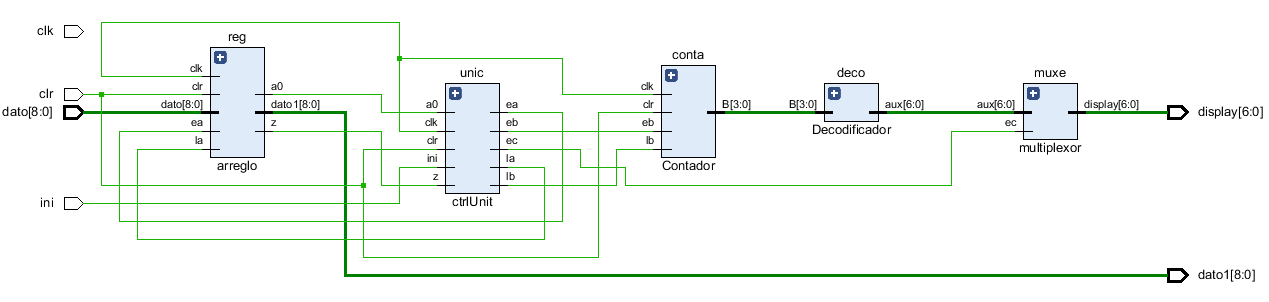
        TbSimEnded <= '1';

        wait;

    end process;

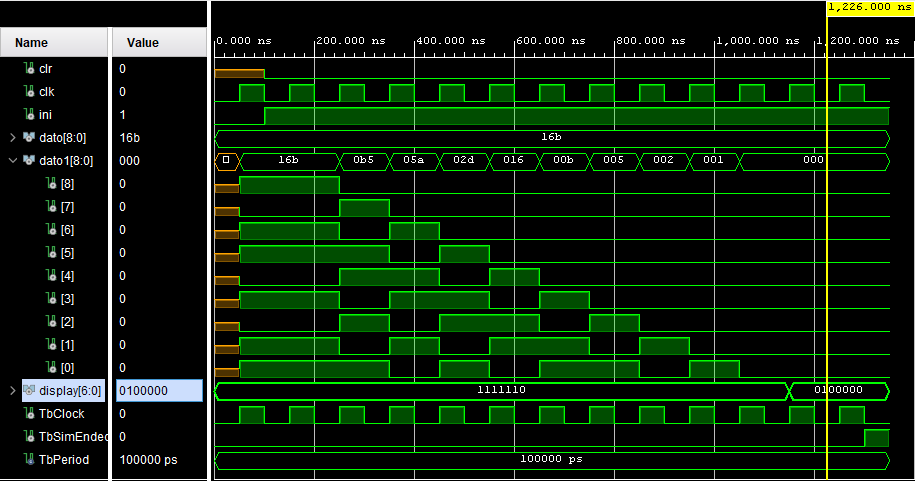
end tb;

### Diagrama RTL





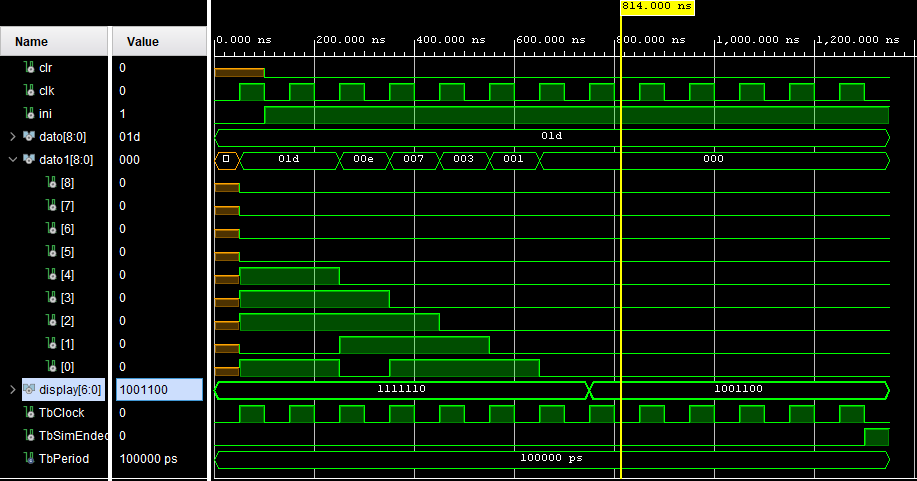
1. 101101011



Observamos como la salida llamada dato 1 va cambiando hasta llegar a ser completamente 0, es decir que podemos observar como se realiza el desplazamiento.

Además, a la salida del display podemos observar el vector “0100000”, y dado que el display es uno de ánodo común, cada 0 indica el segmento encendido, teniendo en este caso a la salida el número 6.

1. 000011101



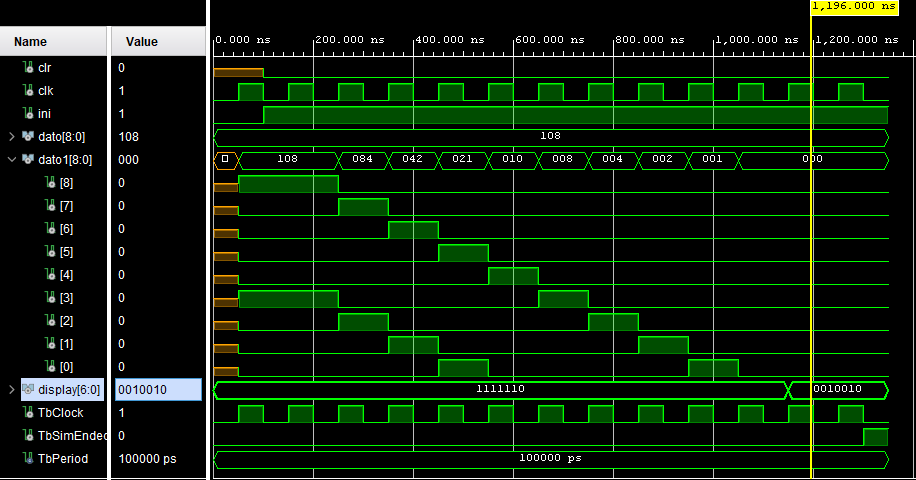
Observamos que la salida del display tiene el vector “1001100”. Dado que trabajamos con display de ánodo común, este seria el equivalente al número 4. El cual corresponde al número de unos existentes en el vector de entrada.

1. 000010000



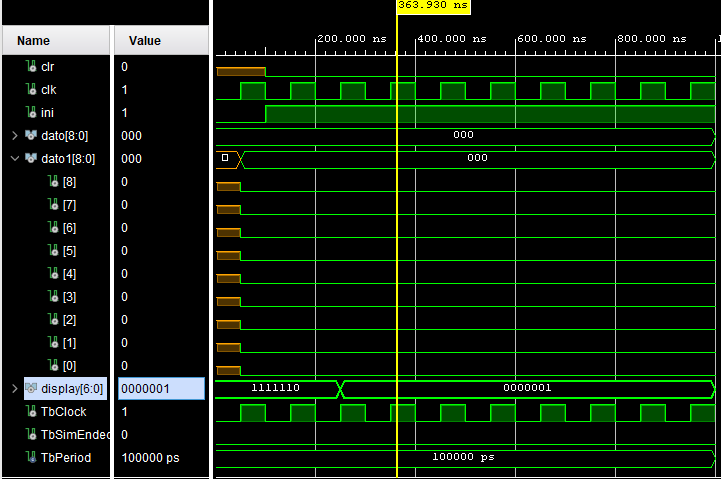
Observamos como el uno se va desplazando a lo largo de la ejecución del programa, además observamos que a la salida obtenemos el vector “1001111”, el cual corresponde a un 1 mostrado en un display de ánodo común.

1. 100001000



Observamos como se realiza el desplazamiento de bits, además a la salida notamos que tenemos el vector “0010010” el cual corresponde a la representación de un numero 2 mostrado en un display de ánodo común.

1. 000000000



Observamos que se verifica inmediatamente que el dato de entrada es 0, por lo cual se obtiene pronto a la salida el vector “0000001”, el cual es la representación de un 0 en un display de ánodo común.