INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

U.A: Arquitectura de Computadoras

“Práctica 11: Cartas ASM”

Grupo: 3CV11

Profesor: Vega García Nayeli

Sánchez Becerra Ernesto Daniel

Unidad de Control

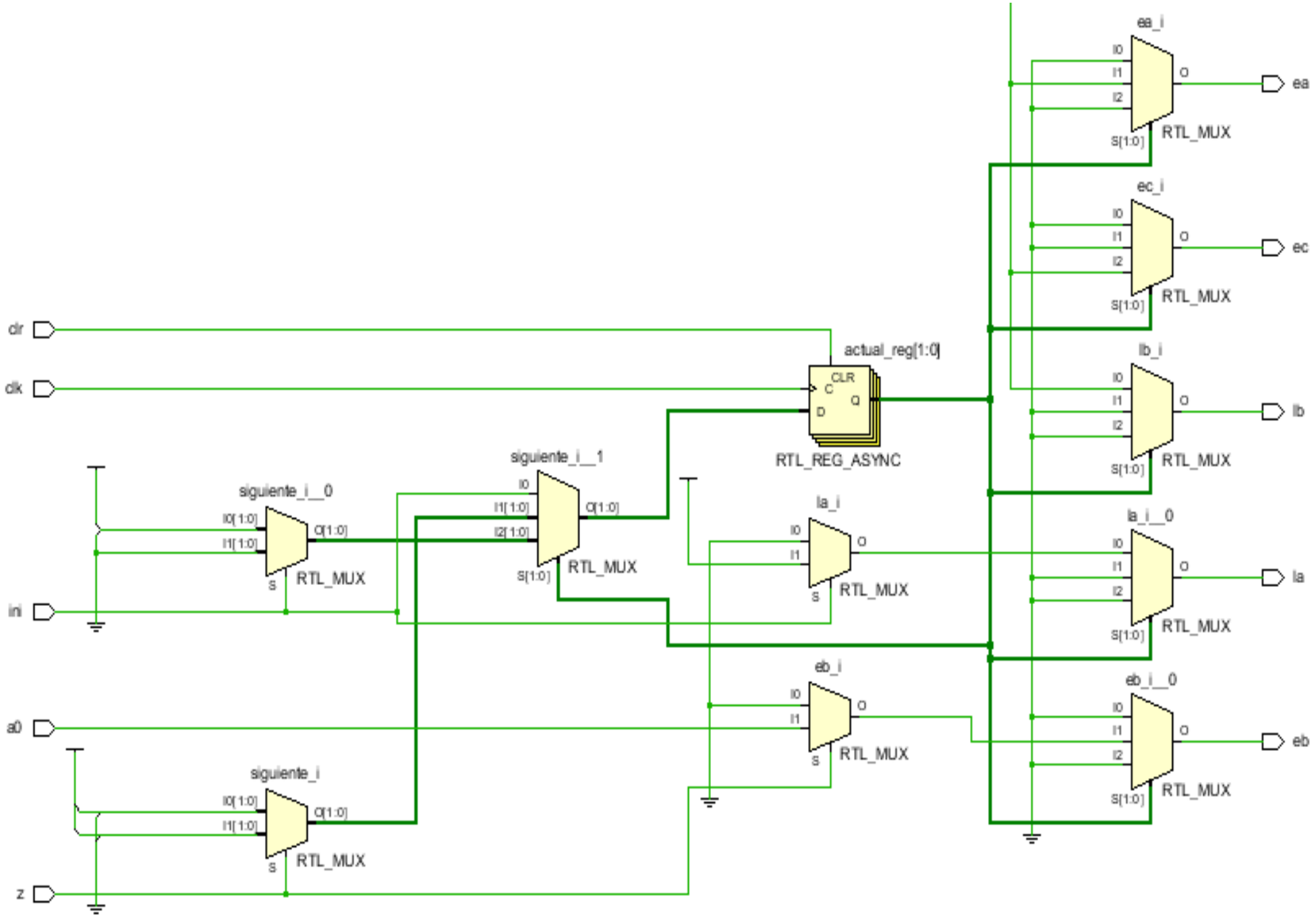
Código de implementación

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity unidad_control is
    Port ( ini : in STD_LOGIC;
           clr : in STD_LOGIC;
           clk : in STD_LOGIC;
           a0 : in STD_LOGIC;
           z : in STD_LOGIC;
           la : out STD_LOGIC;
           ea : out STD_LOGIC;
           lb : out STD_LOGIC;
           ec : out STD_LOGIC;
           eb : out STD_LOGIC);
end unidad_control;

architecture Behavioral of unidad_control is

    type estados is (e0,e1,e2);
    signal actual,siguiente : estados;

    begin

    process(clk, clr)
        begin
        if (clr = '1') then
            actual <= e0;
        elsif (rising_edge(clk)) then
            actual <= siguiente;
        end if;

    end process;

    process(actual,ini,z,a0)
        begin
        la <= '0';
        lb <= '0';
        ea <= '0';
        eb <= '0';
        ec <= '0';

        case actual is
            when e0 =>
                lb <= '1';
                if (ini = '1') then
                    siguiente <= e1;
                else
                    la <= '1';
                    siguiente <= e0;
                end if;
            when e1 =>
                ea <= '1';
                if (z = '1') then -- arreglo igual a cero
                    siguiente <= e2;
                else -- arreglo diferente de cero
                    if (a0 = '1') then
                        eb <= '1';
                        siguiente <= e1;
                    else
                        siguiente <= e1;
                    end if;
                end if;
            when e2 =>
                ec <= '1';
                if (ini = '1') then
                    siguiente <= e2;
                else
                    siguiente <= e0;
                end if;
        end case;

    end process;

end Behavioral;
```

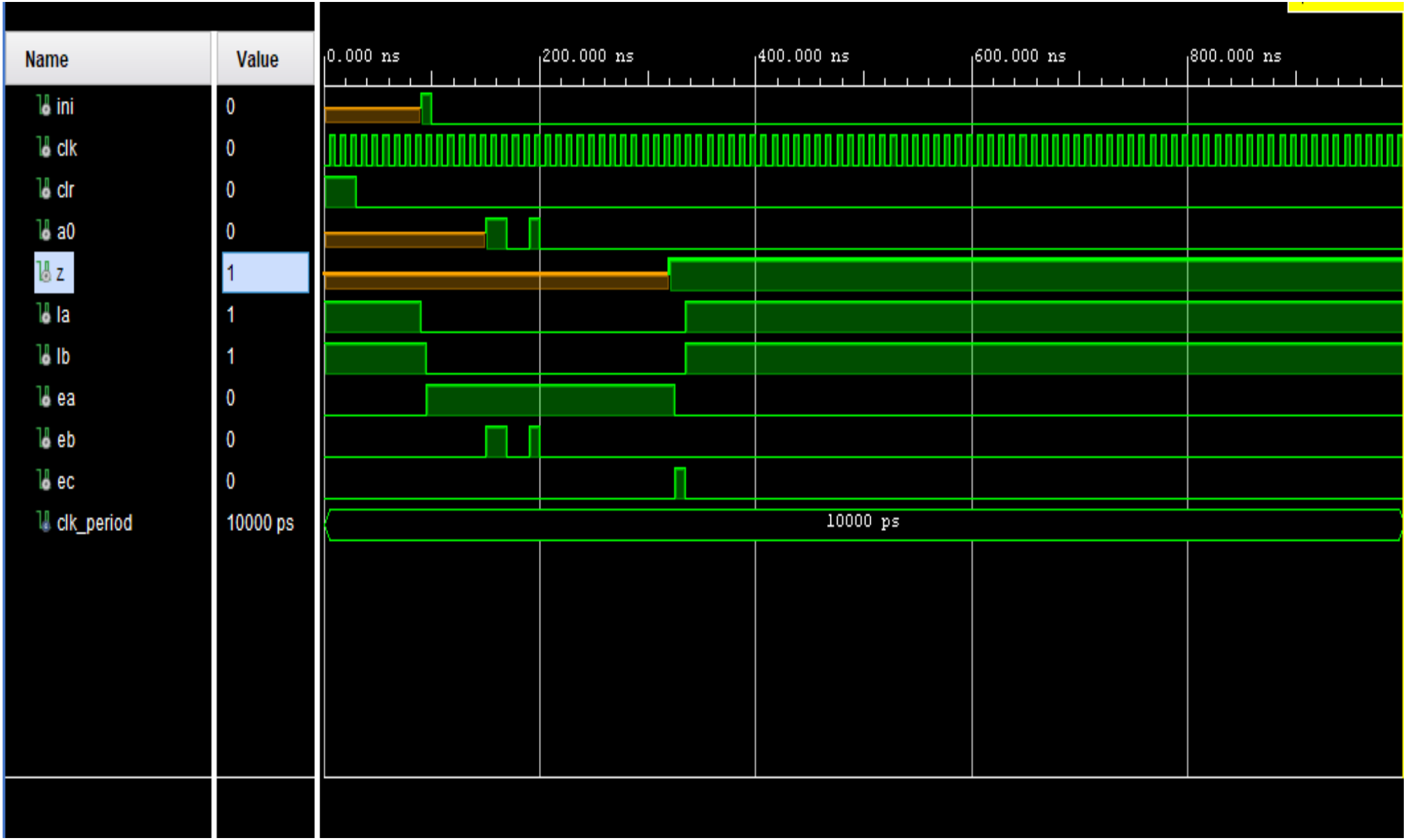
Código de simulación

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_unidad_control is
end tb_unidad_control;

architecture Behavioral of tb_unidad_control is
    component unidad_control
    Port ( ini : in STD_LOGIC;
           clr : in STD_LOGIC;
           clk : in STD_LOGIC;
           a0 : in STD_LOGIC;
           z : in STD_LOGIC;
           la : out STD_LOGIC;
           ea : out STD_LOGIC;
           lb : out STD_LOGIC;
           ec : out STD_LOGIC;
           eb : out STD_LOGIC);
    end component;
    -- input signs
    signal ini, clk, clr, a0, z : STD_LOGIC;

    -- output signs
    signal la, lb, ea, eb, ec : STD_LOGIC;

    -- Clock period
    constant clk_period : time := 10ns;

    begin
    -- Instanciate
    UC : unidad_control Port map (
           ini => ini,
           clr => clr,
           clk => clk,
           a0 => a0,
           z => z,
           la => la,
           ea => ea,
           lb => lb,
           ec => ec,
           eb => eb
    );

    -- Clock process
    CLOCK : process
        begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;

    -- process stimulli
    SP : process
        begin
        clr <= '1';
        wait for 30 ns;
        clr <= '0';
        wait for 60 ns;
        ini <= '1';
        wait for 10 ns;
        ini <= '0';
        wait for 50 ns;
        a0 <= '1';
        wait for 20 ns;
        a0 <= '0';
        wait for 20 ns;
        a0 <= '1';
        wait for 10 ns;
        a0 <= '0';
        wait for 120 ns;
        z <= '1';
        wait;
    end process;


end Behavioral;
```

Diagrama RTL



Forma de onda



Contador

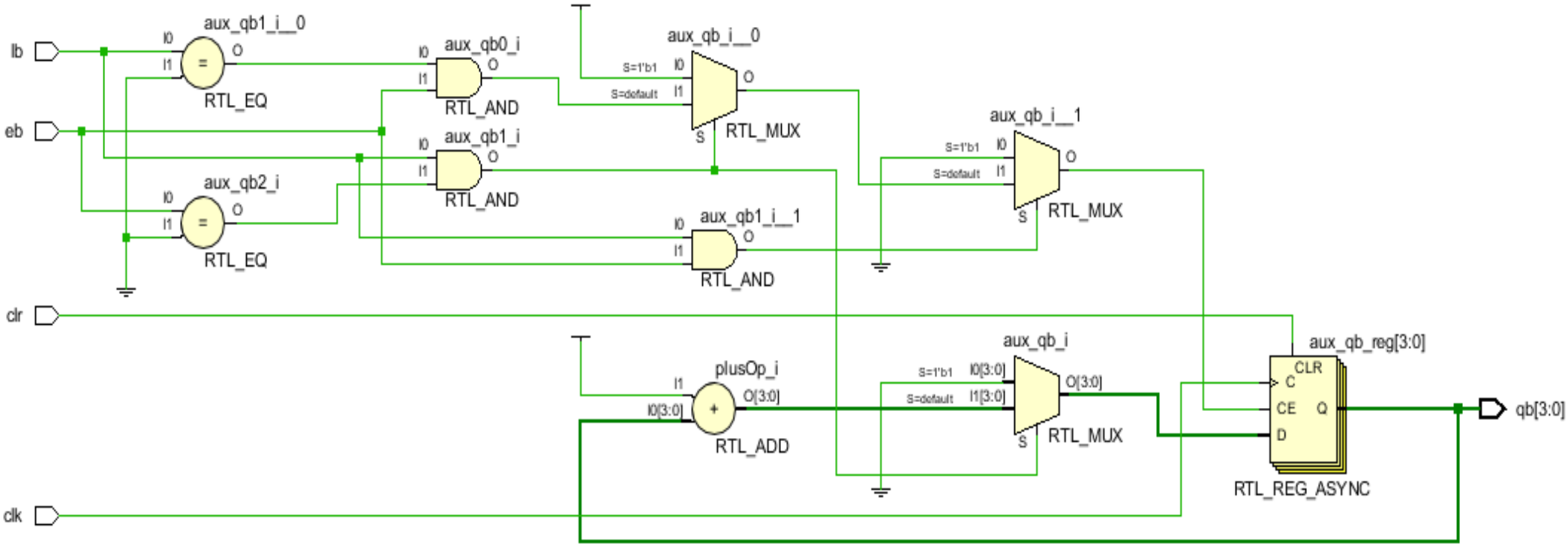
Código de implementación

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Contador is
    Port ( lb : in STD_LOGIC;
           eb : in STD_LOGIC;
           clr : in STD_LOGIC;
           clk : in STD_LOGIC;
           qb : out STD_LOGIC_VECTOR (3 downto 0));
end Contador;

architecture Behavioral of Contador is
    constant zero: std_logic_vector(3 downto 0) := "0000";
    begin
        process(clk, clr, eb, lb)
            variable aux_qb : STD_LOGIC_VECTOR(3 downto 0);
            begin
            if (clr = '1') then
                aux_qb := zero;
            elsif (rising_edge(clk)) then
                if (lb='1' and eb='1') then
                    aux_qb := aux_qb;
                elsif (lb='1' and eb='0') then
                    aux_qb := zero;
                elsif (lb='0' and eb='1') then
                    aux_qb := aux_qb + 1;
                end if;
            end if;

            qb <= aux_qb;
        end process;

end Behavioral;
```

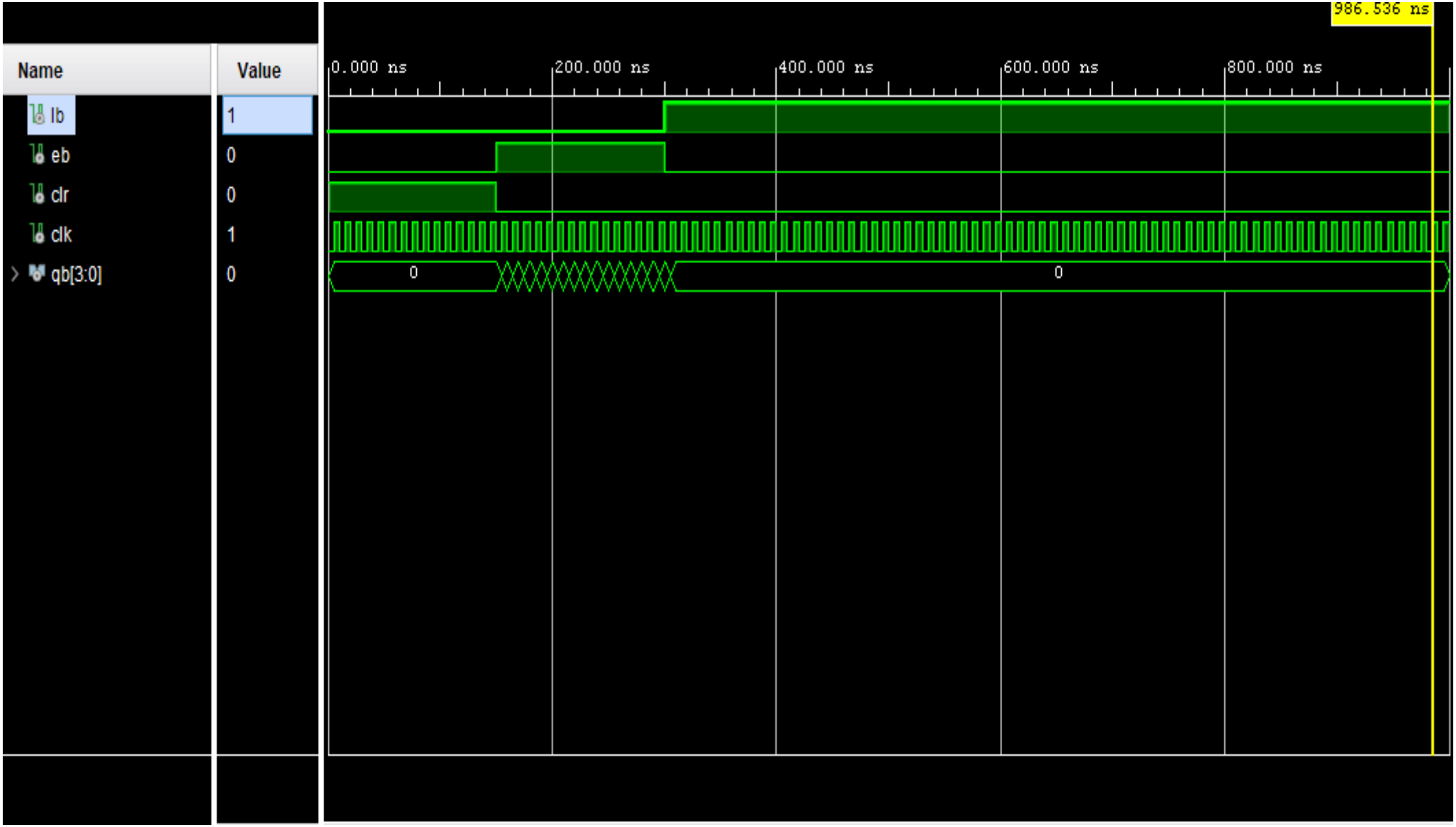
Código de simulación

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_Contador is
end tb_Contador;

architecture Behavioral of tb_Contador is
    component Contador is
    Port ( lb : in STD_LOGIC;
           eb : in STD_LOGIC;
           clr : in STD_LOGIC;
           clk : in STD_LOGIC;
           qb : out STD_LOGIC_VECTOR (3 downto 0));
    end component;

    -- input signs
    signal lb, eb, clr, clk : STD_LOGIC;

    -- output signs
    signal qb : STD_LOGIC_VECTOR(3 downto 0);

    begin

    -- Instaciate component for test
    COUNT : Contador Port map(
           lb => lb,
           eb => eb,
           clr => clr,
           clk => clk,
           qb => qb
    );

    -- Clock process
    CLOCK : process
        begin
        clk <= '0';
        wait for 5ns;
        clk <= '1';
        wait for 5ns;
    end process;

    -- Stimulus process
    SP : process
        begin
        clr <= '1';
        lb <= '0';
        eb <= '0';
        wait for 150 ns;

        clr <= '0';
        eb <= '1';
        wait for 150 ns;

        eb <= '0';
        lb <= '1';
        wait for 150 ns;

        wait;
    end process;



end Behavioral;
```

Diagrama RTL



Forma de onda



Arreglo

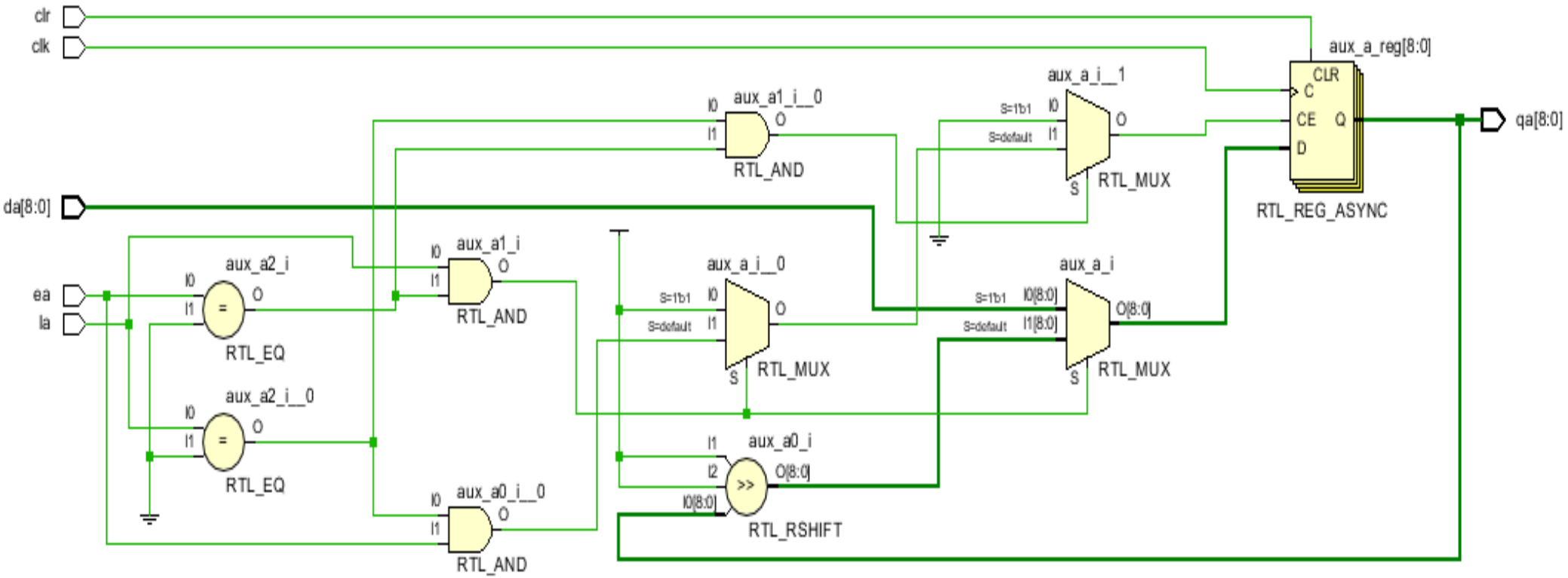
Código de implementación

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Registro is
    Port ( la : in STD_LOGIC;
           ea : in STD_LOGIC;
           clk : in STD_LOGIC;
           clr : in STD_LOGIC;
           da : in STD_LOGIC_VECTOR (8 downto 0);
           qa : out STD_LOGIC_VECTOR (8 downto 0));
end Registro;

architecture Behavioral of Registro is
    signal aux_a : STD_LOGIC_VECTOR(8 downto 0); -- bus
    begin

    process(clk,clr,la,ea)
        begin
        if (clr = '1') then
            aux_a <= "000000000";
        elsif(rising_edge(clk)) then
            if (la = '0' and ea = '0') then
                aux_a <= aux_a;
            elsif (la = '1' and ea = '0') then
                aux_a <= da;
            elsif (la = '0' and ea = '1') then
                aux_a <= to_stdlogicvector(to_bitvector(aux_a) SRL 1);
            end  if;
        end if;
    end process;
    qa <= aux_a;
end Behavioral;
```

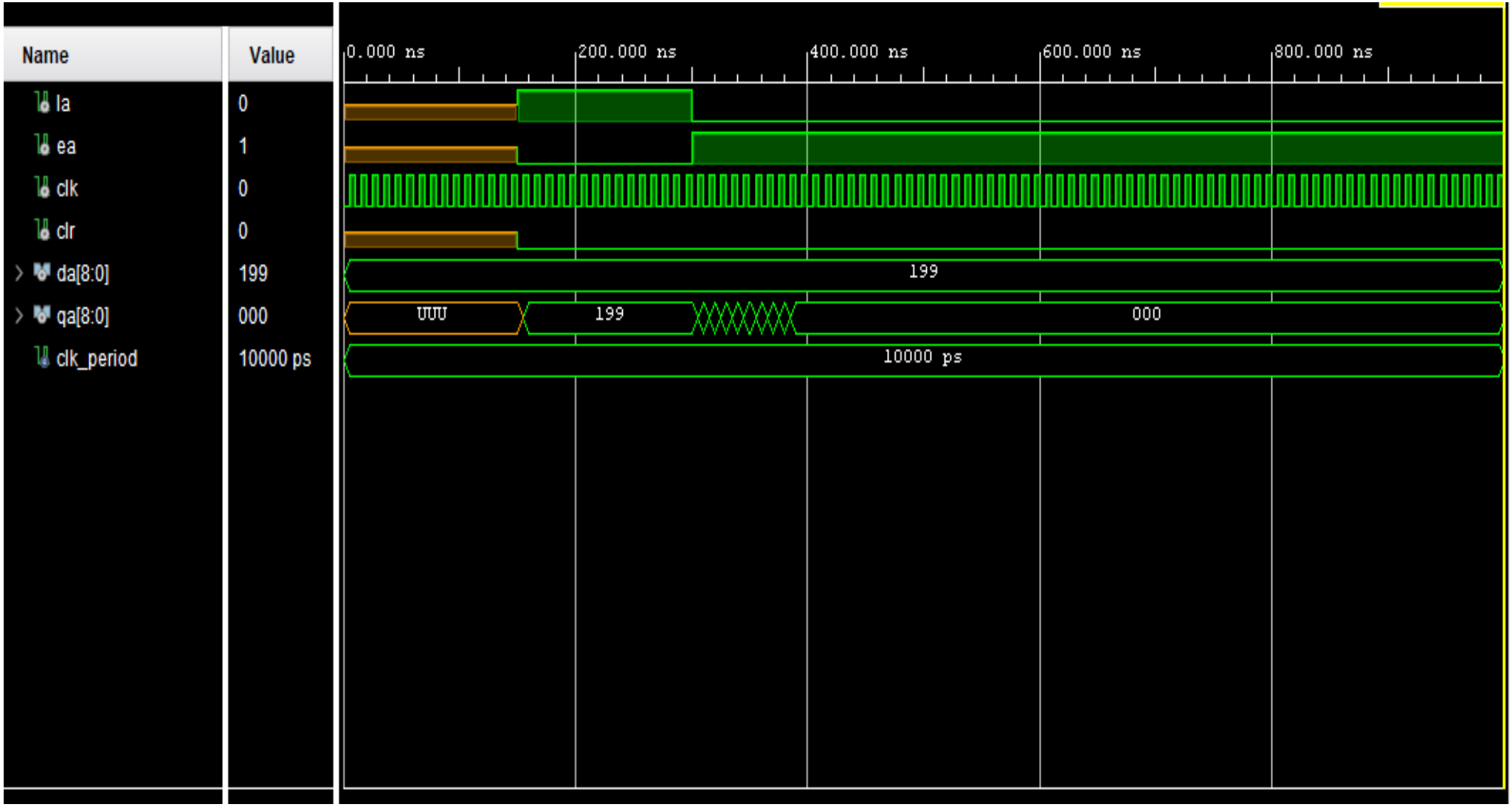
Código de simulación

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_Registro is
end tb_Registro;

architecture Behavioral of tb_Registro is
    component Registro is
    Port ( la : in STD_LOGIC;
           ea : in STD_LOGIC;
           clk : in STD_LOGIC;
           clr : in STD_LOGIC;
           da : in STD_LOGIC_VECTOR (8 downto 0);
           qa : out STD_LOGIC_VECTOR (8 downto 0));
    end component;

    -- input signs
    signal la, ea, clk, clr : STD_LOGIC;
    signal da : STD_LOGIC_VECTOR(8 downto 0);

    -- output signs
    signal qa : STD_LOGIC_VECTOR(8 downto 0);

    -- clock period definition
    constant clk_period : time := 10 ns;

    begin

    -- Instaciate component
    REG : Registro Port map(
           la => la,
           ea => ea,
           clk => clk,
           clr => clr,
           da => da,
           qa => qa
    );

    -- clock process
    CLOCK : process
        begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;

    -- Stimulus process
    SP : process
        begin
        da <= "110011001";
        wait for 150 ns;
        clr <= '0';
        la <= '1';
        ea <= '0';
        wait for 150 ns;
        la <= '0';
        ea <= '1';
        wait for 150 ns;



        wait;
    end process;
end Behavioral;
```

Diagrama RTL



Forma de onda



Decodificador

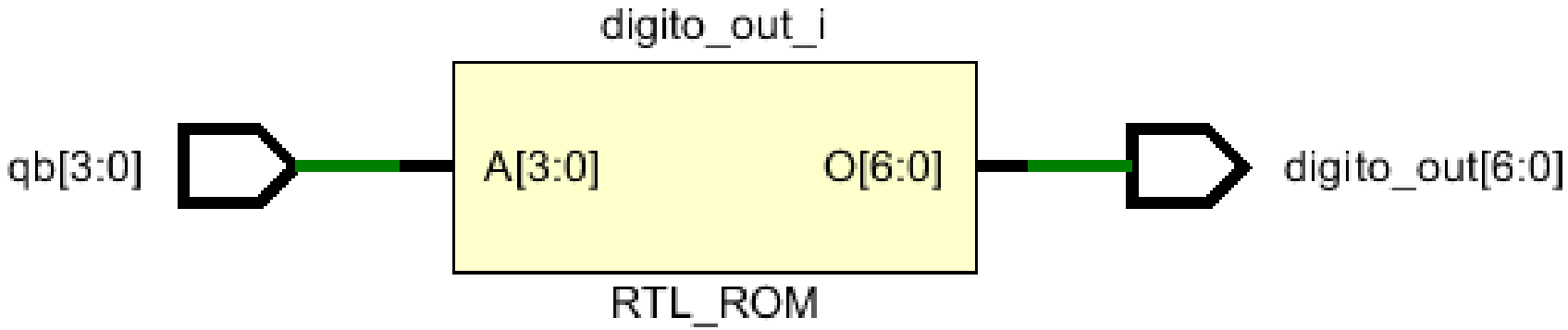
Código de implementación

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Decodificador is
    Port ( qb : in STD_LOGIC_VECTOR (3 downto 0);
           digito_out : out STD_LOGIC_VECTOR (6 downto 0));
end Decodificador;

architecture Behavioral of Decodificador is -- "gfedcba"
    constant digito_0 : STD_LOGIC_VECTOR(6 downto 0) := "0111111";
    constant digito_1 : STD_LOGIC_VECTOR(6 downto 0) := "0000110";
    constant digito_2 : STD_LOGIC_VECTOR(6 downto 0) := "1011011";
    constant digito_3 : STD_LOGIC_VECTOR(6 downto 0) := "1001111";
    constant digito_4 : STD_LOGIC_VECTOR(6 downto 0) := "1100110";
    constant digito_5 : STD_LOGIC_VECTOR(6 downto 0) := "1101101";
    constant digito_6 : STD_LOGIC_VECTOR(6 downto 0) := "1111101";
    constant digito_7 : STD_LOGIC_VECTOR(6 downto 0) := "0000111";
    constant digito_8 : STD_LOGIC_VECTOR(6 downto 0) := "1111111";
    constant digito_9 : STD_LOGIC_VECTOR(6 downto 0) := "1101111";

    begin
    process(qb)
        begin
        case qb is
            when "0000" => digito_out <= digito_0;
            when "0001" => digito_out <= digito_1;
            when "0010" => digito_out <= digito_2;
            when "0011" => digito_out <= digito_3;
            when "0100" => digito_out <= digito_4;
            when "0101" => digito_out <= digito_5;
            when "0110" => digito_out <= digito_6;
            when "0111" => digito_out <= digito_7;
            when "1000" => digito_out <= digito_8;
            when OTHERS => digito_out <= digito_9;
        end case;
    end process;
end Behavioral;
```

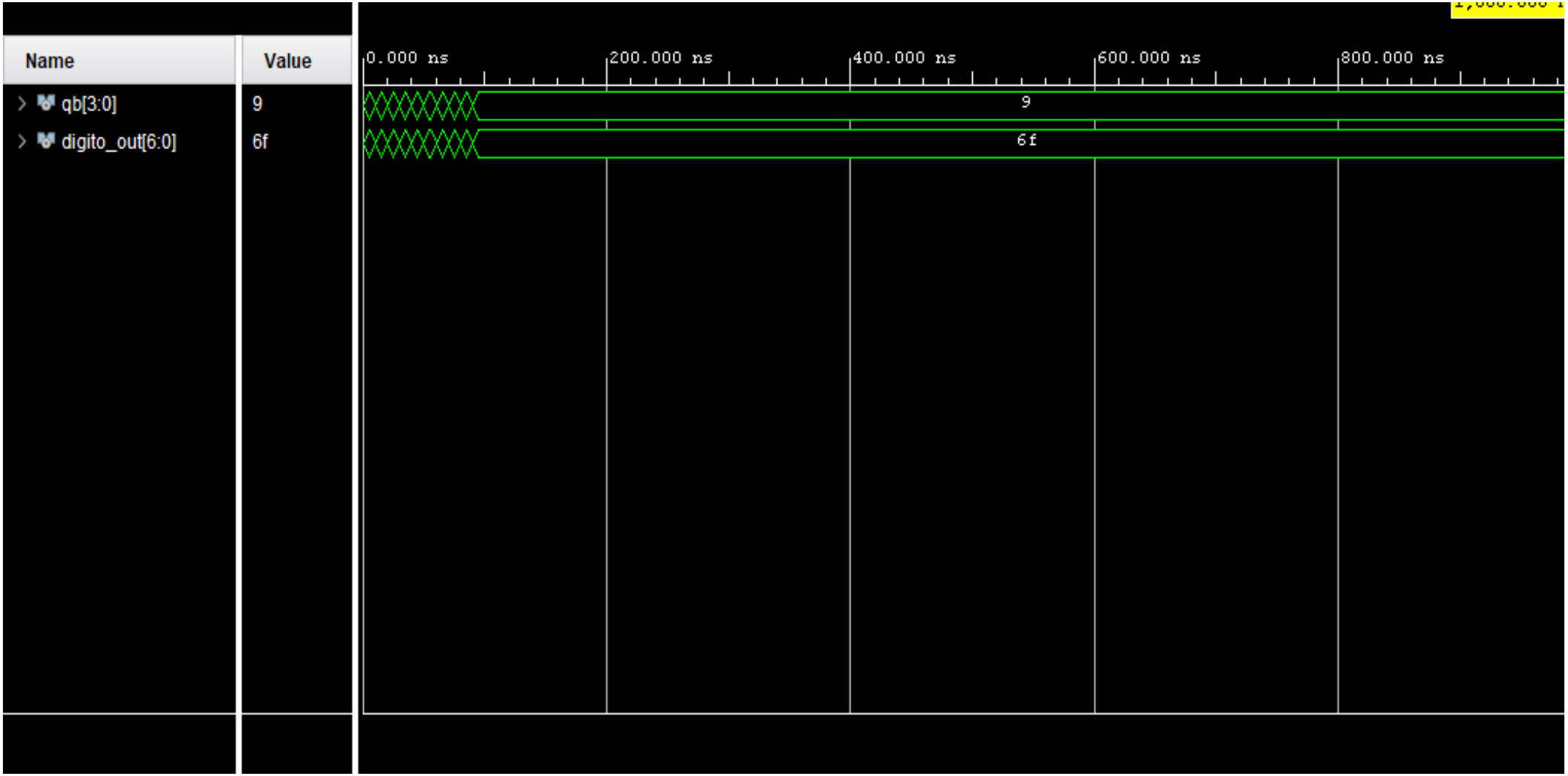
Código de simulación

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_Decodificador is
end tb_Decodificador;

architecture Behavioral of tb_Decodificador is
    component Decodificador is
    Port ( qb : in STD_LOGIC_VECTOR (3 downto 0);
           digito_out : out STD_LOGIC_VECTOR (6 downto 0));
    end component;

    -- input signal
    signal qb : STD_LOGIC_VECTOR (3 downto 0);

    -- output signal
    signal digito_out : STD_LOGIC_VECTOR (6 downto 0);

    begin

    -- Insatnciate compornent
    DECO : Decodificador Port map(
           qb => qb,
           digito_out => digito_out
    );

    -- Stimulus process
    SP : process
        begin
        -- Vamos a probar con cada numero
        qb <= "0000"; -- 0
        wait for 10 ns;
        qb <= "0001"; -- 1
        wait for 10 ns;
        qb <= "0010"; -- 2
        wait for 10 ns;
        qb <= "0011"; -- 3
        wait for 10 ns;
        qb <= "0100"; -- 4
        wait for 10 ns;
        qb <= "0101"; -- 5
        wait for 10 ns;
        qb <= "0110"; -- 6
        wait for 10 ns;
        qb <= "0111"; -- 7
        wait for 10 ns;
        qb <= "1000"; -- 8
        wait for 10 ns;
        qb <= "1001"; -- 9

        wait;
    end process;


end Behavioral;
```

Diagrama RTL

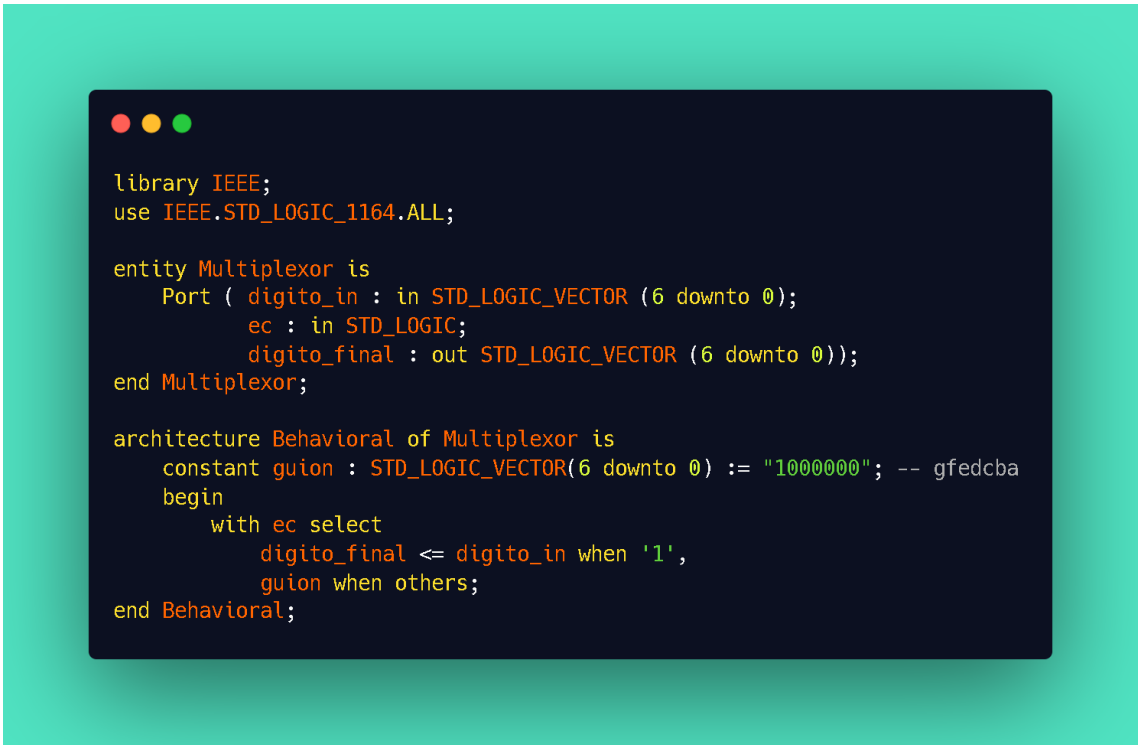


Forma de onda



Multiplexor

Código de implementación

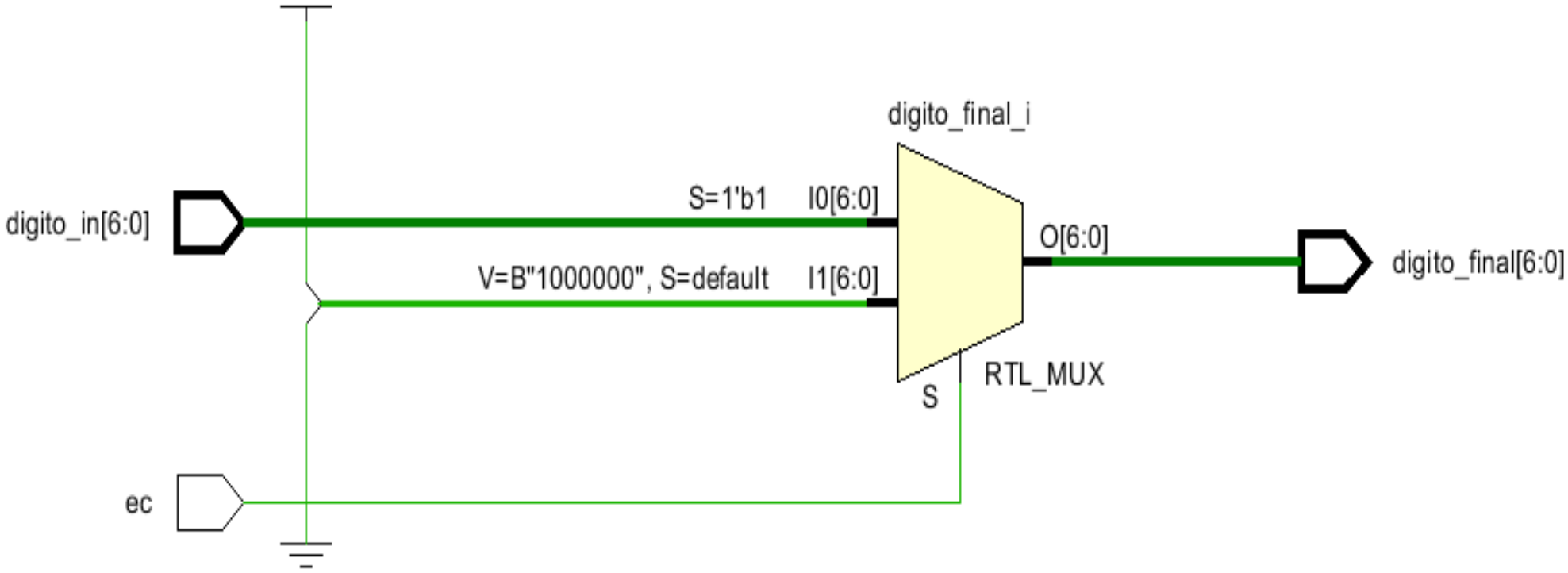


```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Multiplexor is
    Port ( digito_in : in STD_LOGIC_VECTOR (6 downto 0);
           ec : in STD_LOGIC;
           digito_final : out STD_LOGIC_VECTOR (6 downto 0));
end Multiplexor;

architecture Behavioral of Multiplexor is
    constant guion : STD_LOGIC_VECTOR(6 downto 0) := "1000000"; -- gfedcba
    begin
        with ec select
            digito_final <= digito_in when '1',
            guion when others;
end Behavioral;
```

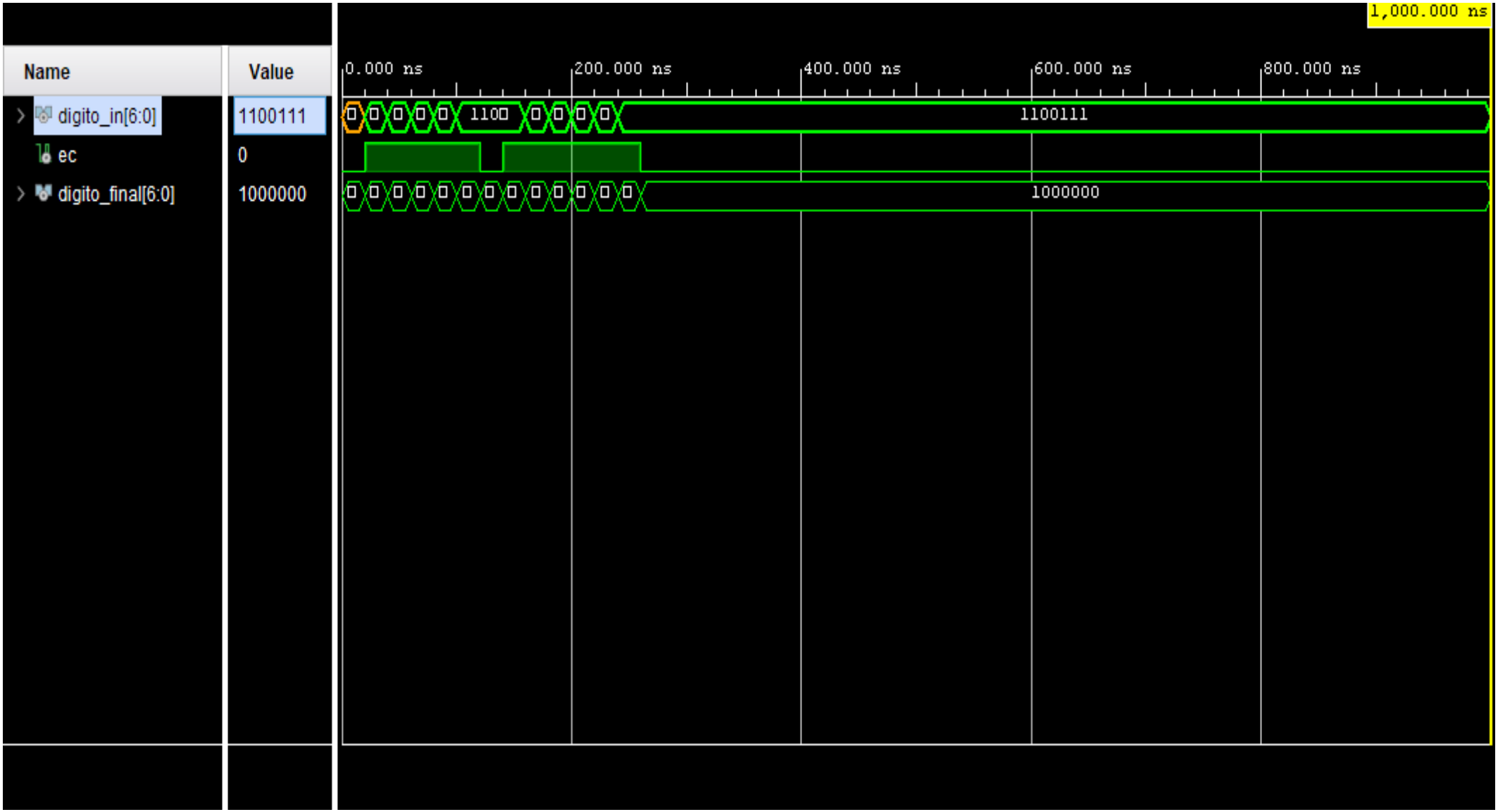
Código de simulación

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_Multiplexor is
end tb_Multiplexor;

architecture Behavioral of tb_Multiplexor is
    component Multiplexor
    Port ( digito_in : in STD_LOGIC_VECTOR (6 downto 0);
           ec : in STD_LOGIC;
           digito_final : out STD_LOGIC_VECTOR (6 downto 0));
    end component;

    -- input signs
    signal digito_in : STD_LOGIC_VECTOR(6 downto 0);
    signal ec : STD_LOGIC;

    -- output sign
    signal digito_final : STD_LOGIC_VECTOR(6 downto 0);

    begin

    -- Instanciate component
    MUX : Multiplexor Port map(
           digito_in => digito_in,
           ec => ec,
           digito_final => digito_final
    );

    -- Stimulus process
    SP : process
        begin
        ec <= '0';
        wait for 20 ns;

        ec <= '1';
        digito_in <= "0111111"; -- 0
        wait for 20 ns;
        digito_in <= "0000110"; -- 1
        wait for 20 ns;
        digito_in <= "1011011"; -- 2
        wait for 20 ns;
        digito_in <= "1001111"; -- 3
        wait for 20 ns;
        digito_in <= "1100110"; -- 4
        wait for 20 ns;
        ec <= '0';
        wait for 20 ns;
        ec <= '1';
        wait for 20 ns;
        digito_in <= "1101101"; -- 5
        wait for 20 ns;
        digito_in <= "1111101"; -- 6
        wait for 20 ns;
        digito_in <= "0000111"; -- 7
        wait for 20 ns;
        digito_in <= "1111111"; -- 8
        wait for 20 ns;
        digito_in <= "1100111"; -- 9
        wait for 20 ns;
        ec <= '0';
        wait for 20 ns;
        wait;
    end process;
end Behavioral;
```

Diagrama RTL



Forma de onda



Arquitectura Completa

Código de implementación

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity cartaASM is
    Port ( clr : in STD_LOGIC;
           clk : in STD_LOGIC;
           ini : in STD_LOGIC;
           data_in : in STD_LOGIC_VECTOR (8 downto 0);
           data_out : out STD_LOGIC_VECTOR (8 downto 0);
           digit_out : out STD_LOGIC_VECTOR (6 downto 0));
end cartaASM;

architecture Behavioral of cartaASM is
    -- unidad de control
    component unidad_control is
    Port ( ini, clr, clk, a0, z : in STD_LOGIC;
           la, ea, lb, ec, eb : out STD_LOGIC);
    end component;

    -- registros
    component Registro is
    Port ( la, ea, clk, clr : in STD_LOGIC;
           da : in STD_LOGIC_VECTOR (8 downto 0);
           qa : out STD_LOGIC_VECTOR (8 downto 0));
    end component;

    -- contador
    component Contador is
    Port ( lb, eb, clr, clk : in STD_LOGIC;
           qb : out STD_LOGIC_VECTOR (3 downto 0));
    end component;

    -- decodificador
    component Decodificador is
    Port ( qb : in STD_LOGIC_VECTOR (3 downto 0);
           digito_out : out STD_LOGIC_VECTOR (6 downto 0));
    end component;

    -- multiplexor
    component Multiplexor is
    Port ( digito_in : in STD_LOGIC_VECTOR (6 downto 0);
           ec : in STD_LOGIC;
           digito_final : out STD_LOGIC_VECTOR (6 downto 0));
    end component;
```

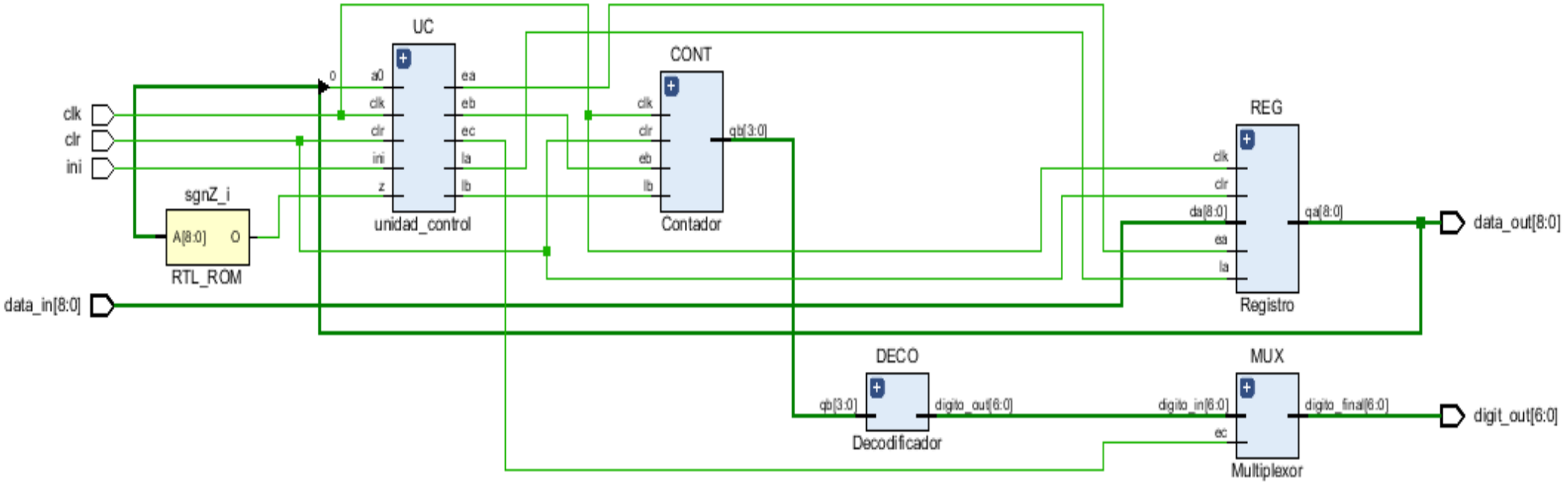
```vhdl
-- señales o buses de alambrado
    signal sgnLA, sgnLB, sgnEA, sgnEB, sgnEC, sgnZ : STD_LOGIC;
    signal sgnOutCounter : STD_LOGIC_VECTOR(3 downto 0);
    signal sgnOutDeco : STD_LOGIC_VECTOR(6 downto 0);
    signal sgnData : STD_LOGIC_VECTOR(8 downto 0);

    begin

    sgnZ <= '1' when sgnData = "000000000" else '0';
    -- Instaciar modulo de unidad de control
    UC : unidad_control
        Port map(
           ini => ini,
           clr => clr,
           clk => clk,
           a0 => sgnData(0),
           z => sgnZ,
           la => sgnLA,
           ea => sgnEA,
           lb => sgnLB,
           ec => sgnEC,
           eb => sgnEB
           );

    -- Instanciar modulo de registro
    REG : Registro
        Port map(
           la => sgnLA,
           ea => sgnEA,
           clk => clk,
           clr => clr,
           da => data_in,
           qa => sgnData
           );

    -- Instanciar modulo de contador
    CONT : Contador
        Port map(
           lb => sgnLB,
           eb => sgnEB,
           clr => clr,
           clk => clk,
           qb => sgnOutCounter
           );

    -- Instanciar modulo de decodificacion
    DECO : Decodificador
        Port map(
           qb => sgnOutCounter,
           digito_out => sgnOutDeco
           );

    -- Instanciar modulo de multiplexor
    MUX : Multiplexor
        Port map(
           digito_in => sgnOutDeco,
           ec => sgnEC,
           digito_final => digit_out
           );

    data_out <= sgnData;

end Behavioral;
```

Código de simulación

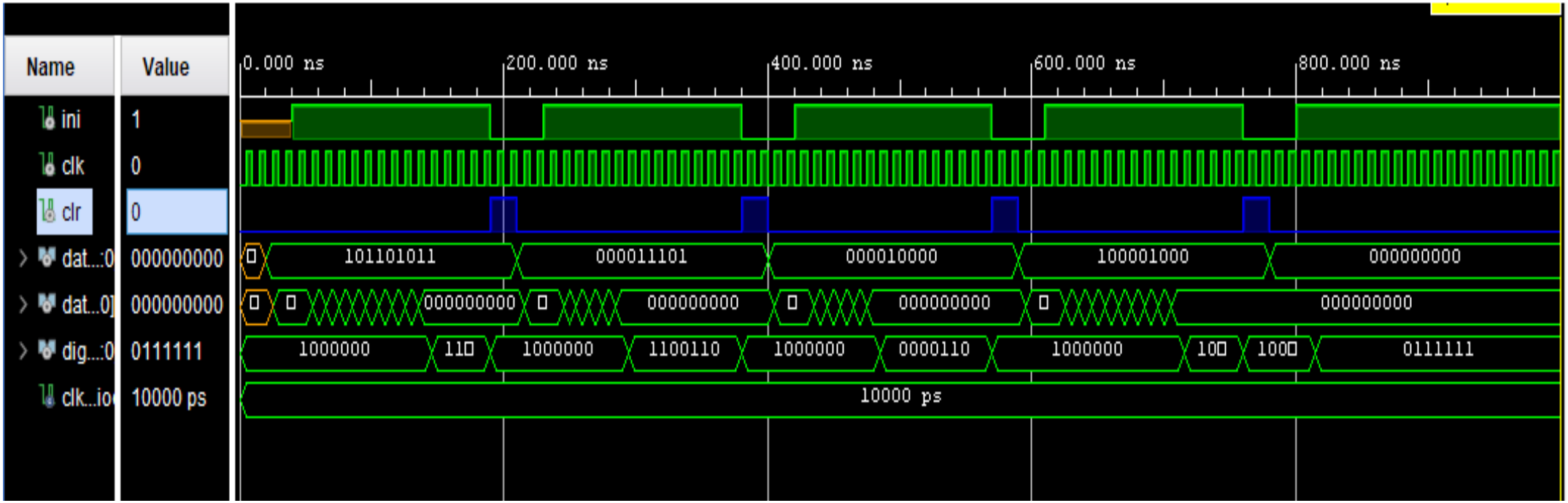
```vhdl
 library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_asm is
end tb_asm;

architecture Behavioral of tb_asm is

    component cartaASM
    Port ( clr : in STD_LOGIC;
           clk : in STD_LOGIC;
           ini : in STD_LOGIC;
           data_in : in STD_LOGIC_VECTOR (8 downto 0);
           data_out : out STD_LOGIC_VECTOR (8 downto 0);
           digit_out : out STD_LOGIC_VECTOR (6 downto 0));
    end component;

    -- input signs
    signal ini : STD_LOGIC;
    signal clk : STD_LOGIC;
    signal clr : STD_LOGIC;
    signal data_in : STD_LOGIC_VECTOR (8 downto 0);

    -- outpuu signs
    signal data_out : STD_LOGIC_VECTOR (8 downto 0);
    signal digit_out : STD_LOGIC_VECTOR (6 downto 0);

    constant clk_period : time := 10 ns;

    begin

    -- clock
    CLOCK_P : process
        begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;

    uut :  cartaASM Port map(
           clr => clr,
           clk => clk,
           ini => ini,
           data_in => data_in,
           data_out => data_out,
           digit_out => digit_out
    );
```

```
-- Process stimulos
   process_stimuli : process
       begin

       clr <= '1';
       clr <= '0';
       wait for 20ns;
       data_in <= "101101011";
       wait for 20 ns;
       ini <= '1';
       wait for 150 ns;

       ini <= '0';
       clr <= '1';
       wait for 20 ns;
       clr <= '0';
       data_in <= "000011101";
       wait for 20 ns;
       ini <= '1';
       wait for 150 ns;

       ini <= '0';
       clr <= '1';
       wait for 20 ns;
       clr <= '0';
       data_in <= "000010000";
       wait for 20 ns;
       ini <= '1';
       wait for 150 ns;

       ini <= '0';
       clr <= '1';
       wait for 20 ns;
       clr <= '0';
       data_in <= "100001000";
       wait for 20 ns;
       ini <= '1';
       wait for 150 ns;

       ini <= '0';
       clr <= '1';
       wait for 20 ns;
       clr <= '0';
       data_in <= "000000000";
       wait for 20 ns;
       ini <= '1';
       wait for 150 ns;

   wait;
   end process;
end Behavioral;
```

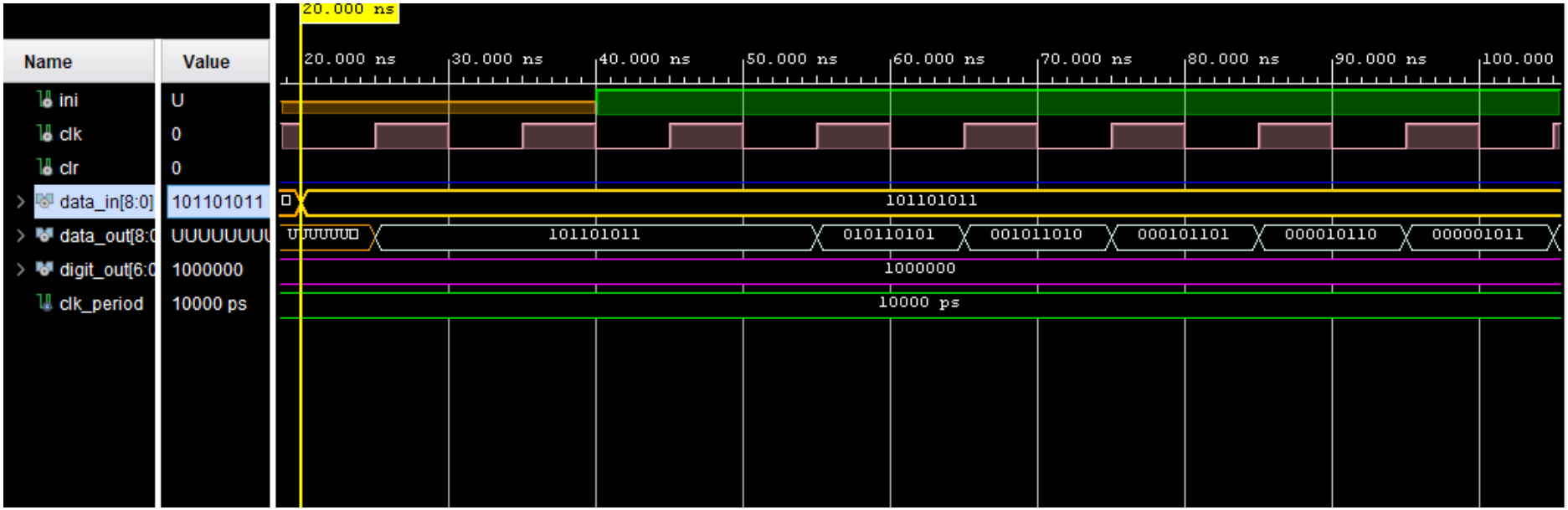
Diagrama RTL



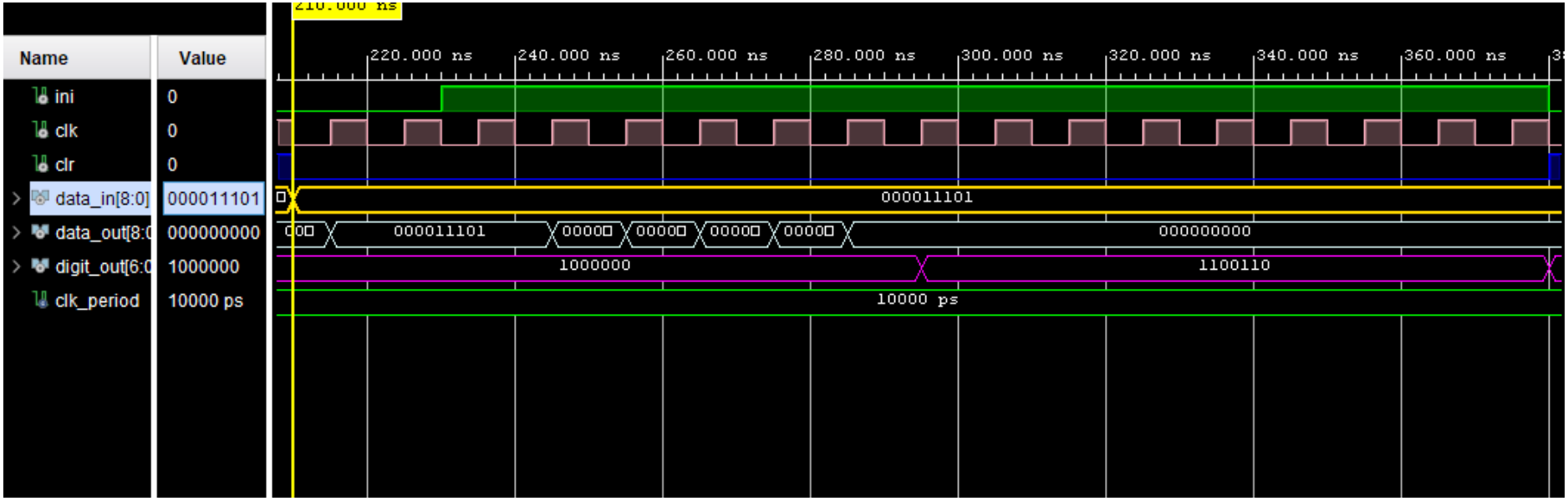
Forma de onda



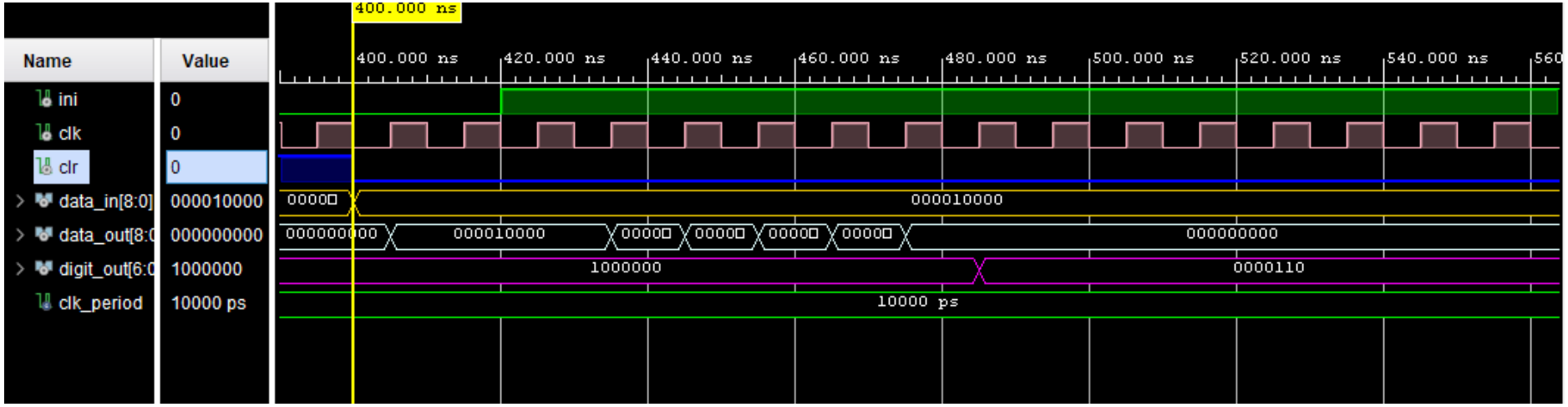
Simulación a



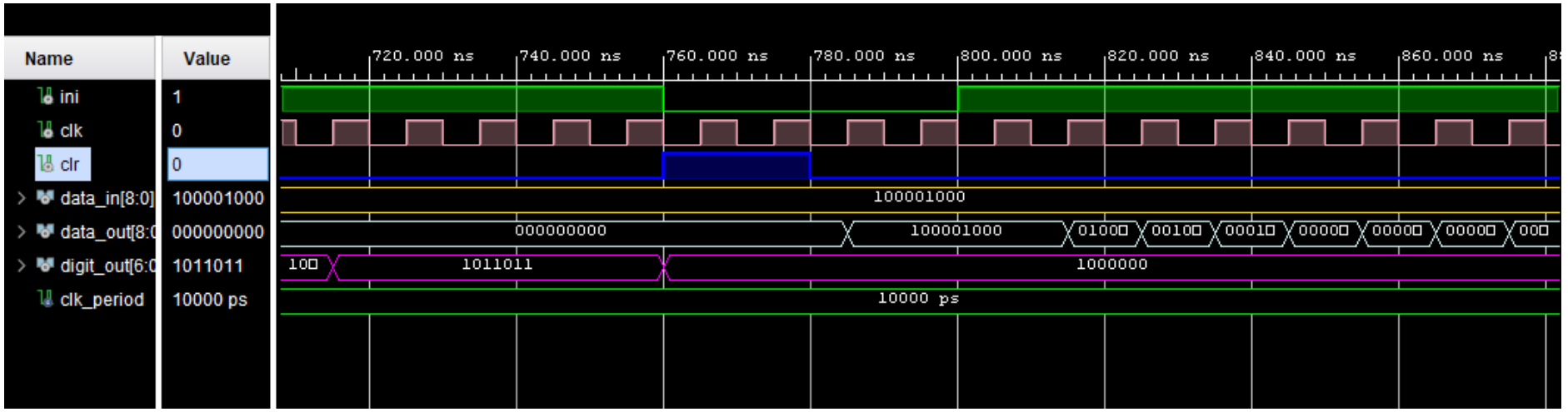
Simulación b



Simulación c



Simulación d



Simulación e