Instituto Politécnico Nacional

Escuela Superior de Computo

Práctica 11.

Cartas ASM.

Nombre: Flores Castro Luis Antonio.

Arquitectura de Computadoras.

Profesora: Vega García Nayeli.

Código de Implementación de la Unidad de Control.

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

entity UC is
        Port(INI, clr, clk, z, A0: in STD_LOGIC;
                LA, EA, LB, EB, EC: out STD_LOGIC);
end UC;

architecture Behavioral of UC is
                type edo is (E0, E1, E2);
                signal estado_act, estado_sig: edo;
        begin

                process(clr, clk)
                begin
                        if(clr='1') then
                                estado_act<=E0;
                        elsif(rising_edge(clk)) then
                                estado_act<=estado_sig;
                        end if;
                end process;

                --aqui se inicia la carta asm
                process(estado_act, INI, z, A0)
                begin
                        LA<='0';
                        EA<='0';
                        LB<='0';
                        EB<='0';
                        EC<='0';

                        case estado_act is
                        when E0=>
                                LB<='1';
                                if(INI='1')then
                                        estado_sig<=E1;
                                else
                                        LA<='1';
                                        estado_sig<=E0;
                                end if;
                        when E1=>
                                EA<='1';
                                if(z='0')then
                                        if(A0='1')then
                                                EB<='1';
                                                estado_sig<=E1;
                                        else
                                                estado_sig<=E1;
                                        end if;
                                        else
                                                estado_sig<=E2;
                                        end if;
                                when E2=>
                                        EC<='1';
                                        if(INI='1')then
                                                estado_sig<=E2;
                                        else
                                                estado_sig<=E0;
                                        end if;
                                end case;
                        end process;
                end Behavioral;
```

Código de Implementación del Arreglo.

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

entity Arreglo 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 Arreglo;

        architecture Behavioral if Arreglo is
                signal auxQA: STD_LOGIC_VECTOR(8 downto 0);
        begin
                process(LA, EA, clk, clr)
                begin
                if(clr='1')then
                        auxQA<=(others=>'0');
                elsif(rising_edge(clk))then
                        if(LA='0' and EA='0')then
                                auxQA<=auxQA;
                        elsif(LA='1' and EA='0')then
                                auxQA<=DA;
                        elsif(LA='0' and EA='1')then
                                auxQA<=to_stdlogicvector(to_bitvector(auxQA)SRL 1);
                        end if;
                end if;
                end process;
                QA<=auxQA;
        end Behavioral;
```

Código de Implementación del Contador.

```
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, EB, clk, clr: in STD_LOGIC;
        QB: out STD_LOGIC_VECTOR(3 downto 0));
end Contador;

        architecture Behavioral of Contador is
        begin
                process(clk,clr,LB,EB)
                variable auxQB: STD_LOGIC_VECTOR(3 downto 0);
                begin
                        if(clr='1')then
                                auxQB:=(others=>'0');
                        elsif(rising_edge(clk))then
                                if(LB='0' and EB='0')then
                                        auxQB:=auxQB;
                                elsif(LB='1' and EB='0')then
                                        auxQB:=(others=>'0');
                                elsif(LB='0' and EB='1')then
                                        auxQB:=auxQB+1;
                                end if;
                        end if;
                        QB<=auxQB;
                end process;
        end Behavioral;
```

Código de Implementación del Decodificador.

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

entity Decodificador is
Port(B: in STD_LOGIC_VECTOR(3 downto 0);
        no: out STD_LOGIC_VECTOR(6 downto 0));
end Decodificador;

architecture Behavioral of Decodificador is
        constant n0: STD_LOGIG_VECTOR(6 downto 0):="0000001";
        constant n1: STD_LOGIG_VECTOR(6 downto 0):="1001111";
        constant n2: STD_LOGIG_VECTOR(6 downto 0):="0010010";
        constant n3: STD_LOGIG_VECTOR(6 downto 0):="0000110";
        constant n4: STD_LOGIG_VECTOR(6 downto 0):="1001100";
        constant n5: STD_LOGIG_VECTOR(6 downto 0):="0100100";
```

```
    constant n6: STD_LOGIG_VECTOR(6 downto 0):="0100000";
    constant n7: STD_LOGIG_VECTOR(6 downto 0):="0001110";
    constant n8: STD_LOGIG_VECTOR(6 downto 0):="0000000";
    constant n9: STD_LOGIG_VECTOR(6 downto 0):="0001100";

    begin
        process(B)
        begin
            case B is
                when "0000"=>no<=n0;
                when "0001"=>no<=n1;
                when "0010"=>no<=n2;
                when "0011"=>no<=n3;
                when "0100"=>no<=n4;
                when "0101"=>no<=n5;
                when "0110"=>no<=n6;
                when "0111"=>no<=n7;
                when "1000"=>no<=n8;
                when others=>no<=n9;
            end case;
        end process;
    end Behavioral;
```

Código de Implementación del Multiplexor.

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

entity Multiplexor is
Port(nod: 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 caracter: STD_LOGIC_VECTOR(6 downto 0):="1111110";
    begin
        with EC select
            display<=num when '1',
            caracter when others;
    end Behavioral;
```

Código de Test Bench de la Unidad de Control.

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

entity TbUC is
end TbUC;

architecture Behavioral of TbUC is

        component UC is
                    Port(INI, clr, clk, z, A0: in STD_LOGIC;
                            LA, EA, LB, EB, EC: out STD_LOGIC);
            end component;

    signal clk, clr INI, z, A0, LA, LB, EA, EB, EC: STD_LOGIC:='0';
    begin

            simu : UC
            Port map(
            clk=>clk,
            clr=>clr,
            INI=>INI,
            z=>z,
            A0=>A0,
            LA=>LA,
            LB=>LB,
            EA=>EA,
            EB=>EB,
            EC=>EC
            );

    clock: process begin
            clk<='0';
            wait for 5 ns;
            clk<='1';
            wait for 5 ns;
    end process;

    process
    begin
            clr<='1';
            wait for 40 ns;
            clr<='0';
            wait for 40 ns;
            INI<='1';
            wait for 40 ns;
            INI<='0';
            wait for 40 ns;
```

```vhdl
             A0<='1';
             wait for 40 ns;
             A0<='0';
             wait for 40 ns;
             A0<='1';
             wait for 40 ns;
             A0<='0';
             wait for 40 ns;
             z<='1';
             wait;
             end process;
     end Behavioral;
```

Código de Test Bench del Arreglo.

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

entity TbArreglo is
end tbArreglo;

architecture Behavioral of TbArreglo is

        component Arreglo 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;

             signal LA, EA, clk, clr: STD_LOGIC:='0';
             signal DA,
QA: STD_LOGIC_VECTOR(8 downto 0):=(others=>'0');

     begin
             arr: Arreglo
             Port map(
             LA=>LA,
             EA=>EA,
             clk=>clk,
             clr=>clr,
             DA=>DA,
             QA=>QA
             );

             reloj: process begin
             clk<='0';
             wait for 5 ns;
             clk<='1';
             wait for 5 ns;
```

```
                end process;

                process
                begin
                clr<='1';
                wait for 10 ns;
                clr<='0';
                DA<='100000000';
                wait for 10 ns;
                LA<='0';
                EA<='0';
                wait for 10 ns;
                LA<='1';
                EA<='0';
                wait for 10 ns;
                LA<='0';
                EA<='1';
                wait;
                end process;
        end Behavioral;
```

Código de Test Bench del Contador.

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

entity TbContador is
end TbContador;

architecture Behavioral of TbContador is

                component Contador is
                Port(LB, EB, clk, clr: in STD_LOGIC;
                        QB: out STD_LOGIC_VECTOR(3 downto 0));
                end component;

                signal clk,clr,LB,EB: STD_LOGIC:='0';
                signal QB: STD_LOGIC_VECTOR(3 downto 0):=(others =>'0');
        begin
                cnt: Contador
                Port map(
                clk=>clk,
                clr=>clr,
                LB=>LB,
                EB=>EB,
                QB=>QB
```

```vhdl
                );

                clock: process begin
                clk<='0';
                wait for 5 ns;
                clk<='1';
                wait for 5 ns;
                end process;

                process
                begin
                clr<='1';
                wait for 10 ns;
                clr<='0';
                wait for 10 ns;
                LB<='1';
                EB<='0';
                wait for 10 ns;
                LB<='0';
                EB<='1';
                wait for 10 ns;
                LB<='1';
                EB<='0';
                wait for 10 ns;
                LB<='0';
                EB<='1';
                wait for 10 ns;
                LB<='0';
                EB<='0';
                wait;
                end process;
        end Behavioral;
```

Código de paquete de la Carta ASM.

```vhdl
library IEEE;
use IEEE.std_logic_1164.All;

package PaqueteASM is

    component UC is
        Port(INI, clr, clk, z, A0: in STD_LOGIC;
                LA, EA, LB, EB, EC: out STD_LOGIC);
        end component;

                component Arreglo 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;

                component Contador is
                Port(LB, EB, clk, clr: in STD_LOGIC;
                        QB: out STD_LOGIC_VECTOR(3 downto 0));
                end component;

                component Multiplexor is
                Port(nod: in STD_LOGIC_VECTOR(6 downto 0);
                        EC: in STD_LOGIC;
                        display: out STD_LOGIC_VECTOR(6 downto 0
));
                end component;

                component Decodificador is
                Port(B: in STD_LOGIC_VECTOR(3 downto 0);
                        no: out STD_LOGIC_VECTOR(6 downto 0));
                end component;

        end package;
```

Código de Implementación de la Carta ASM.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.PaqueteASM.all;

entity CartaASM is
Port(clk, clr, INI: in STD_LOGIC;
        D: in STD_LOGIC_VECTOR(8 downto 0);
        Dout: out STD_LOGIC_VECTOR(8 downto 0);
        noc: out STD_LOGIC_VECTOR(3 downto 0);
             displayc: out STD_LOGIC_VECTOR(6 downto 0));
     end CartaASM;

     architecture Behavioral of CartaASM is


             signal auxZ, auxLA, auxLB, auxEA, auxEB, auxEC: STD_LOGIC:='0';
             signal auxA: STD_LOGIC_VECTOR(8 downto 0):=(others=>'0');
             signal auxB: STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
             signal auxNo: STD_LOGIC_VECTOR(6 downto 0):=(others=>'0');

     begin
             arr: Arreglo
             Port map(
             LA=>auxLA,
             EA=>auxEA,
             clk=>clk,
             clr=>clr,
             DA=>D,
             QA=>auxA
             );

             auxZ<='1' when auxA="000000000" else '0';

             unidadc: UC
             Port map(
             clk=>clk,
             clr=>clr,
             INI=>INI,
             z=>auxZ,
             A0=>auxA(0),
             LA=>auxLA,
             LB=>auxLB,
             EA=>auxEA,
             EB=>auxEB,
             EC=>auxEC
```

```vhdl
		);

		cnt: Contador
		Port map(
		clk=>clk,
		clr=>clr,
		LB=>auxLB,
		EB=>auxEB,
		QB=>auxB
		);

		dec: Decodificador
		Port map(
		B=>auxB,
		no=>auxNo
		);

		mux: Multiplexor
		Port map(
		nod=>auxNo,
		EC=>auxEC,
		display=>displayc
		);

		noc<=auxB;
		Dout<=auxA;
	end Behavioral;
```

Código de Test Bench de la Carta ASM.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.PaqueteASM.all;

entity TbCartaASM is
end TbCartaASM;

architecture Behavioral of TbCartaASM is

                component CartaASM is
                Port(clk, clr, INI: in STD_LOGIC;
                        D: in STD_LOGIC_VECTOR(8 downto 0);
                        Dout: out STD_LOGIC_VECTOR(8 downto 0);
                        noc: out STD_LOGIC_VECTOR(3 downto 0);
                        displayc: out STD_LOGIC_VECTOR(6 downto 0));
                end component;

                signal clk,clr,INI: STD_LOGIC;
                signal D, Dout: STD_LOGIC_VECTOR(8 downto 0);
                signal displayc: STD_LOGIC_VECTOR(6 downto 0);
                signal noc: STD_LOGIC_VECTOR(3 downto 0);

        begin

                CASM: CartaASM
                Port map(
                clk=>clk,
                clr=>clr,
                INI=>INI,
                D=>D,
                Dout=>Dout,
                noc=>noc,
                displayc=>displayc
                );

                clock: process begin
                clk<='0';
                wait for 5 ns;
                clk<='1';
                wait for 5 ns;
                end process;

                process
                begin
                INI<='0';
                clr<='1';
                wait for 10 ns;

                clr<='0';
                D<="101101011";
                wait for 10 ns;

                INI<='1';
                wait for 110 ns;

                INI<='0';
                clr<='1';
                wait for 10 ns;

                clr<='0';
                D<="000011101";
                wait for 10 ns;

                INI<='1';
                wait for 100 ns;

                INI<='0';
                clr<='1';
                wait for 10 ns;

                clr<='0';
                D<="000010000";
                wait for 10 ns;

                INI<='1';
                wait for 100 ns;

                INI<='0';
                clr<='1';
                wait for 10 ns;

                clr<='0';
                D<="100001000";
                wait for 10 ns;

                INI<='1';
                wait for 100 ns;

                INI<='0';
                clr<='1';
                wait for 10 ns;

                clr<='0';
                D<="000000000";
                wait for 10 ns;

                INI<='1';
                wait;
        end process;
        end Behavioral;
```

Diagrama RTL de la Unidad de Control.

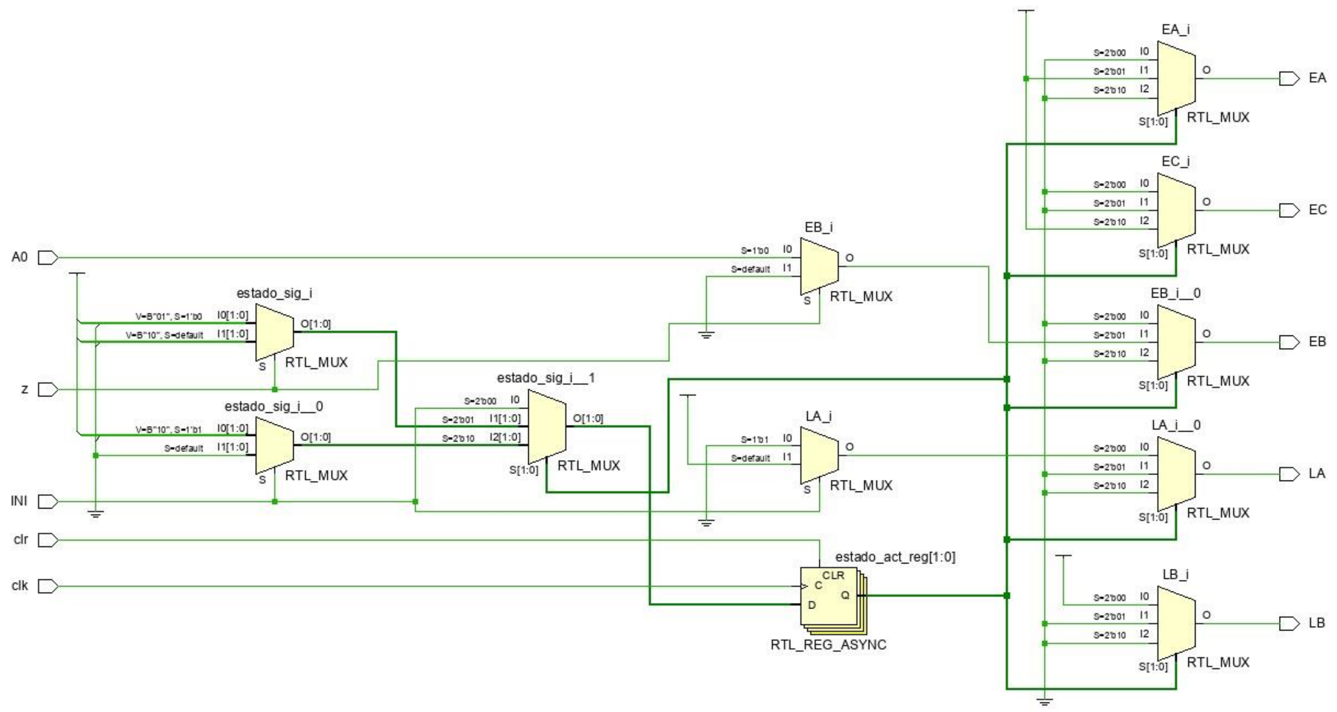

Figura 1. Diagrama RTL de la Unidad de Control.

Diagrama RTL de Arreglo.

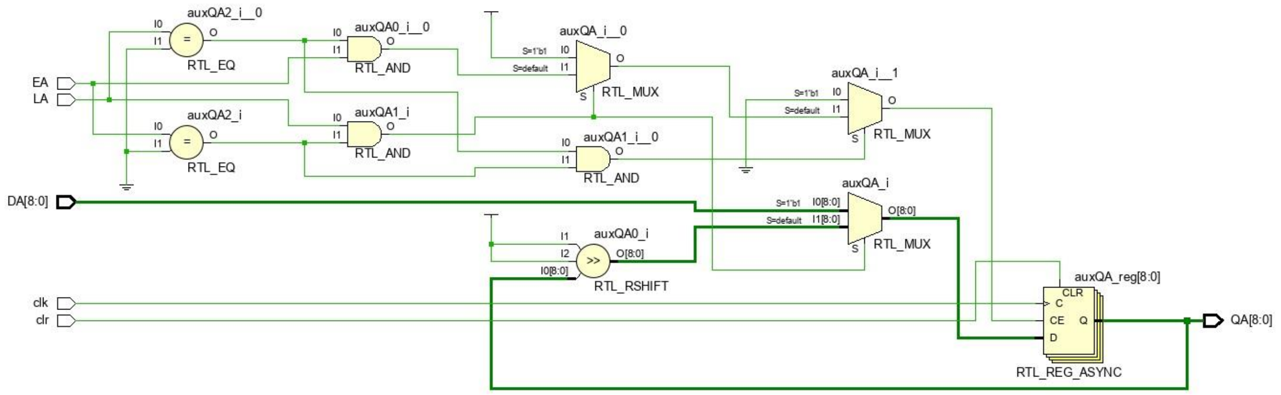

Figura 1.1 Diagrama RTL del Arreglo.

Diagrama RTL de Contador.

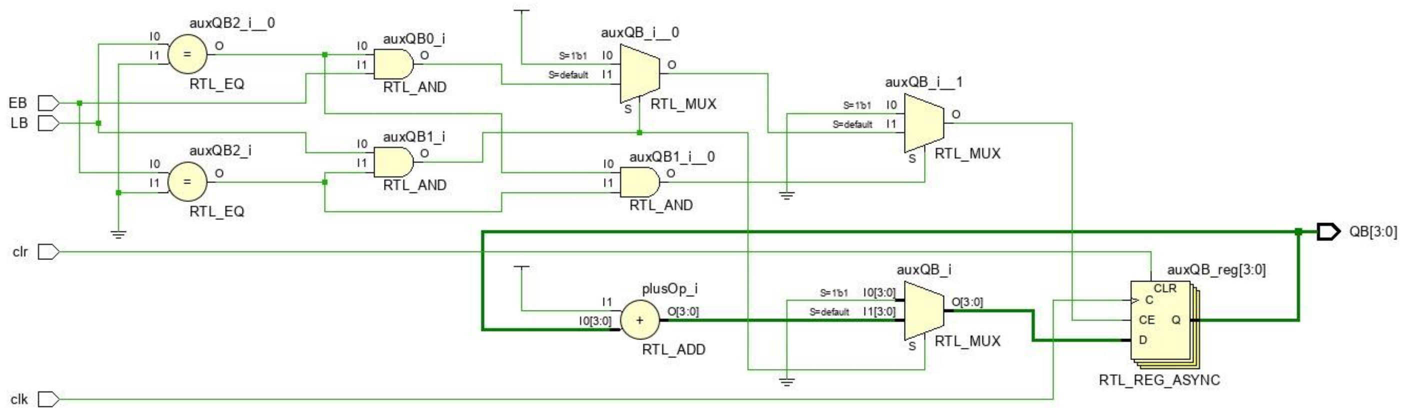

Figura 1.2 Diagrama RTL del Contador.

Diagrama RTL del Decodificador.

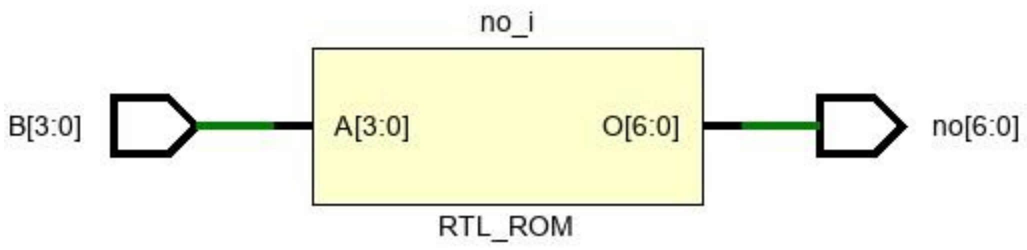

Figura 1.3 Diagrama RTL del Decodificador.

Diagrama RTL del Multiplexor.

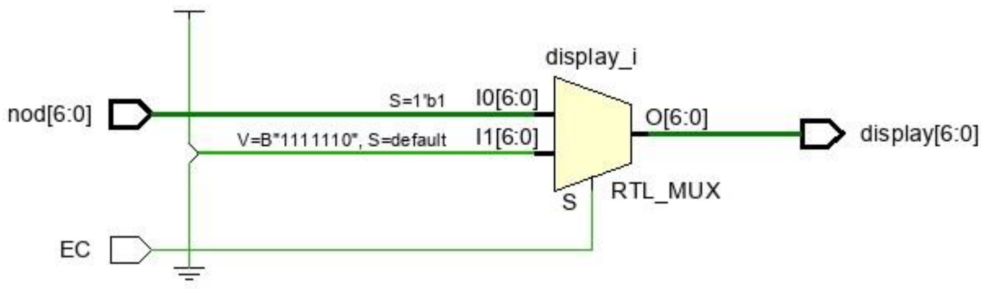

Figura 1.4 Diagrama RTL del Multiplexor.

Diagrama RTL de Carta ASM.

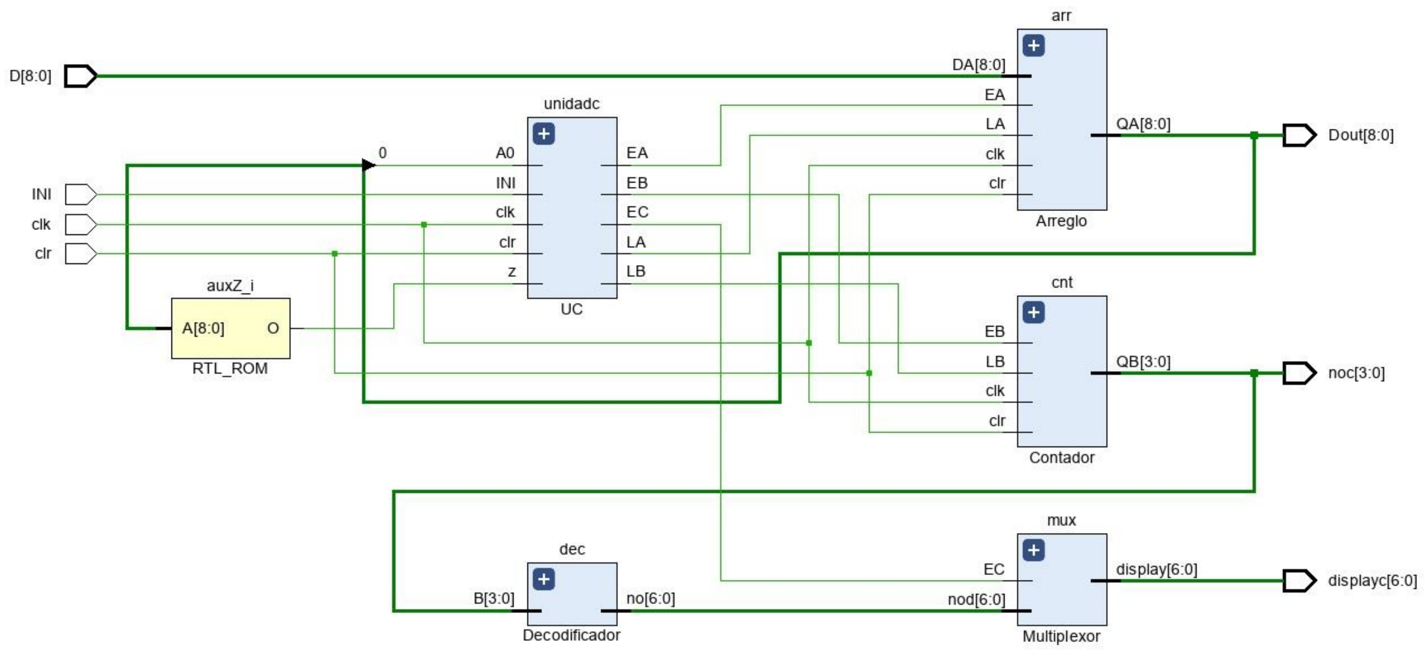

Figura 1.5 Diagrama RTL de la Carta ASM

Diagrama lógico de Unidad de Control.

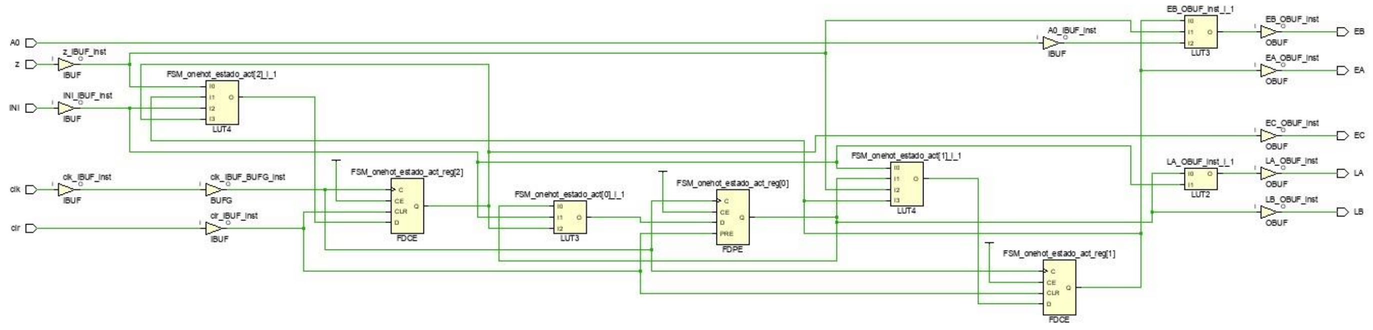

Figura 2 Diagrama lógico de la Unidad de Control.

Diagrama lógico del Arreglo.

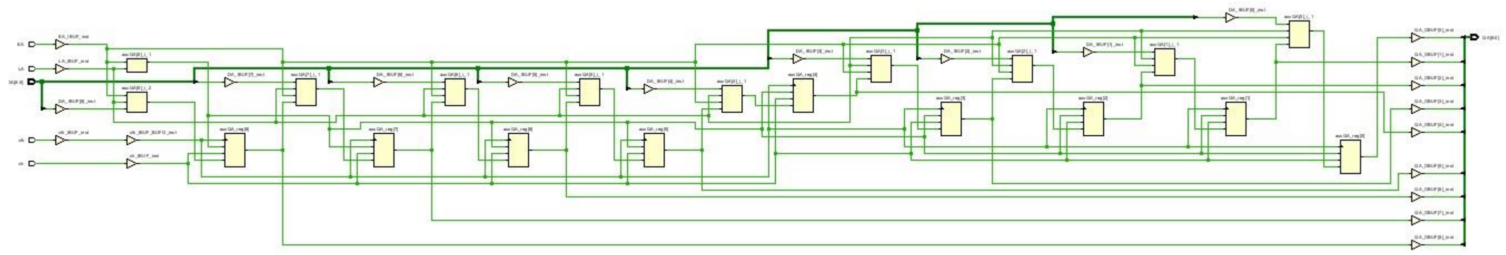

Figura 2.1 Diagrama lógico del Arreglo.

Diagrama lógico del Contador.

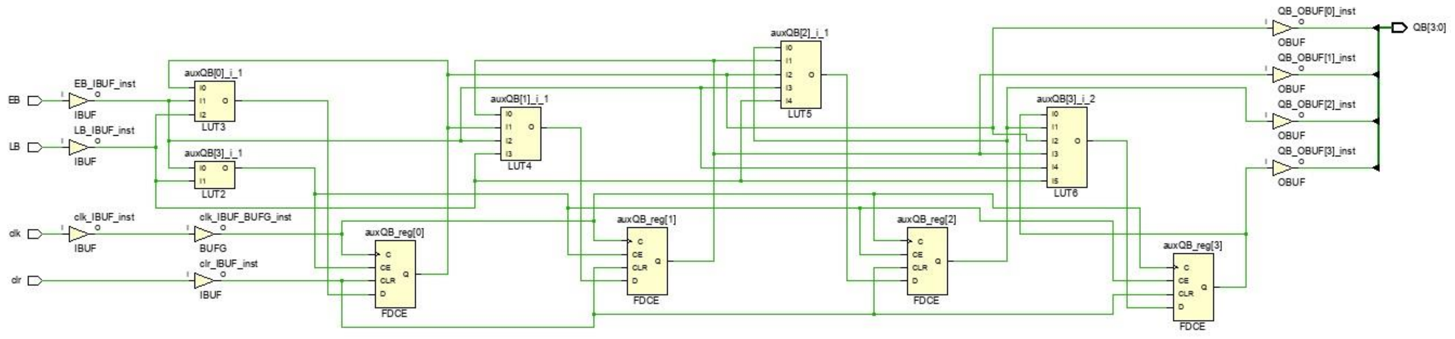

Figura 2.3 Diagrama lógico del Contador.

Diagrama lógico del Decodificador.

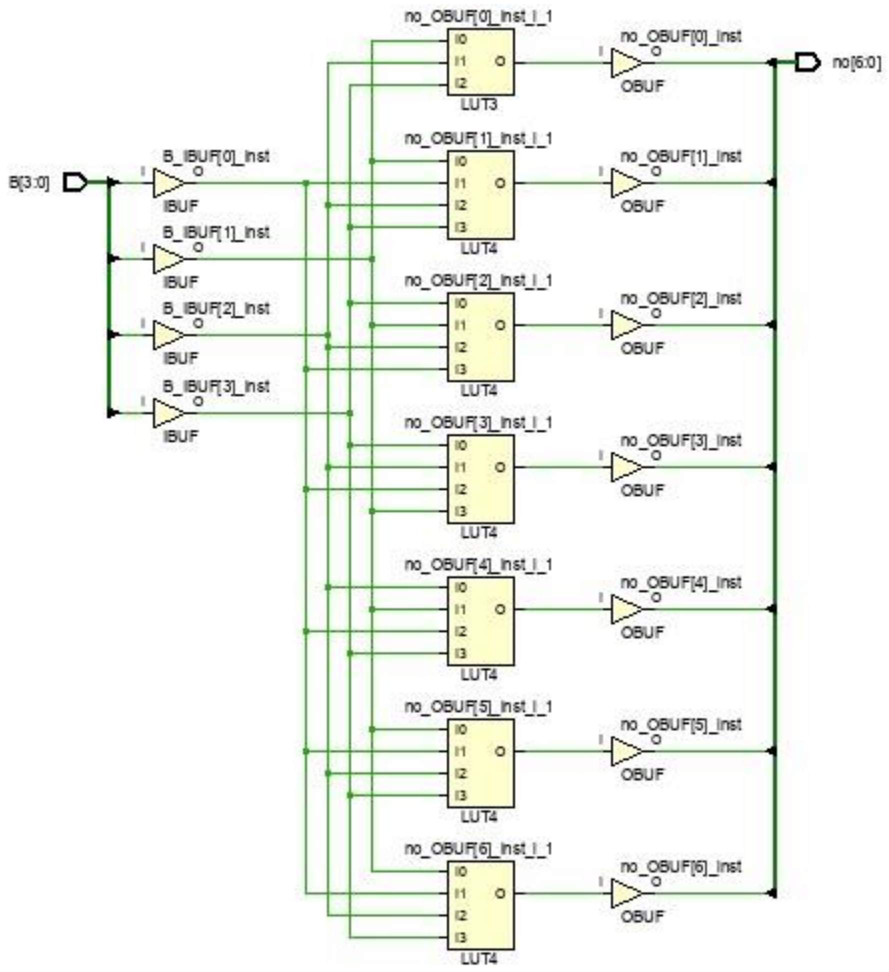

Figura 2.4 Diagrama lógico del Decodificador.

Diagrama lógico del Multiplexor.

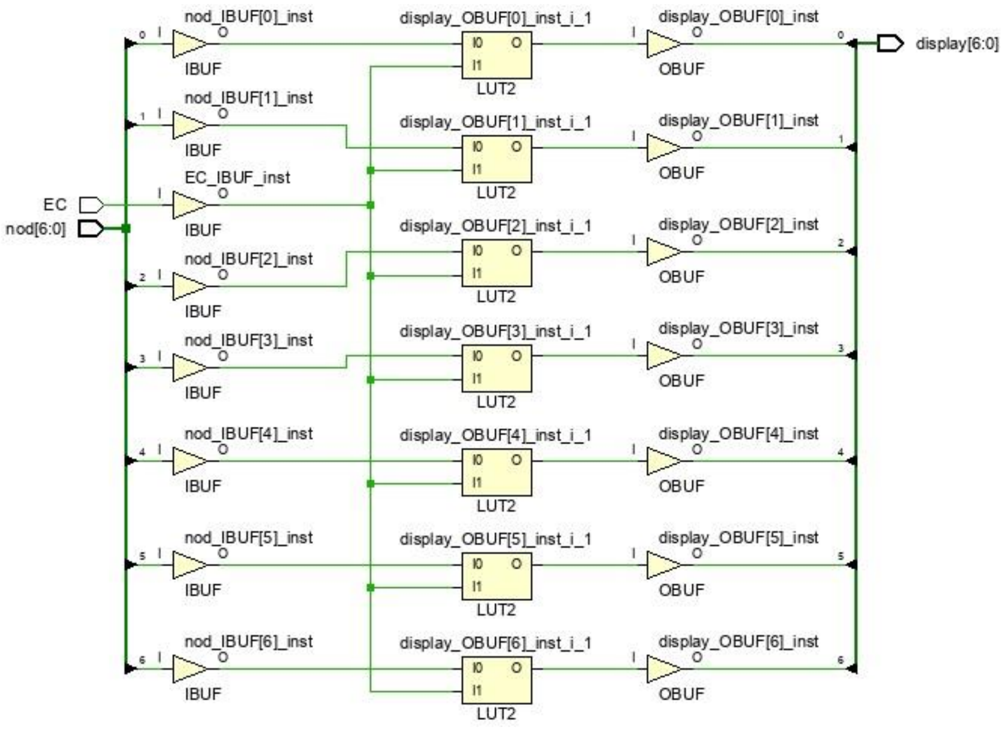

Figura 2.5 Diagrama lógico del Multiplexor.

Diagrama lógico de la Carta ASM.

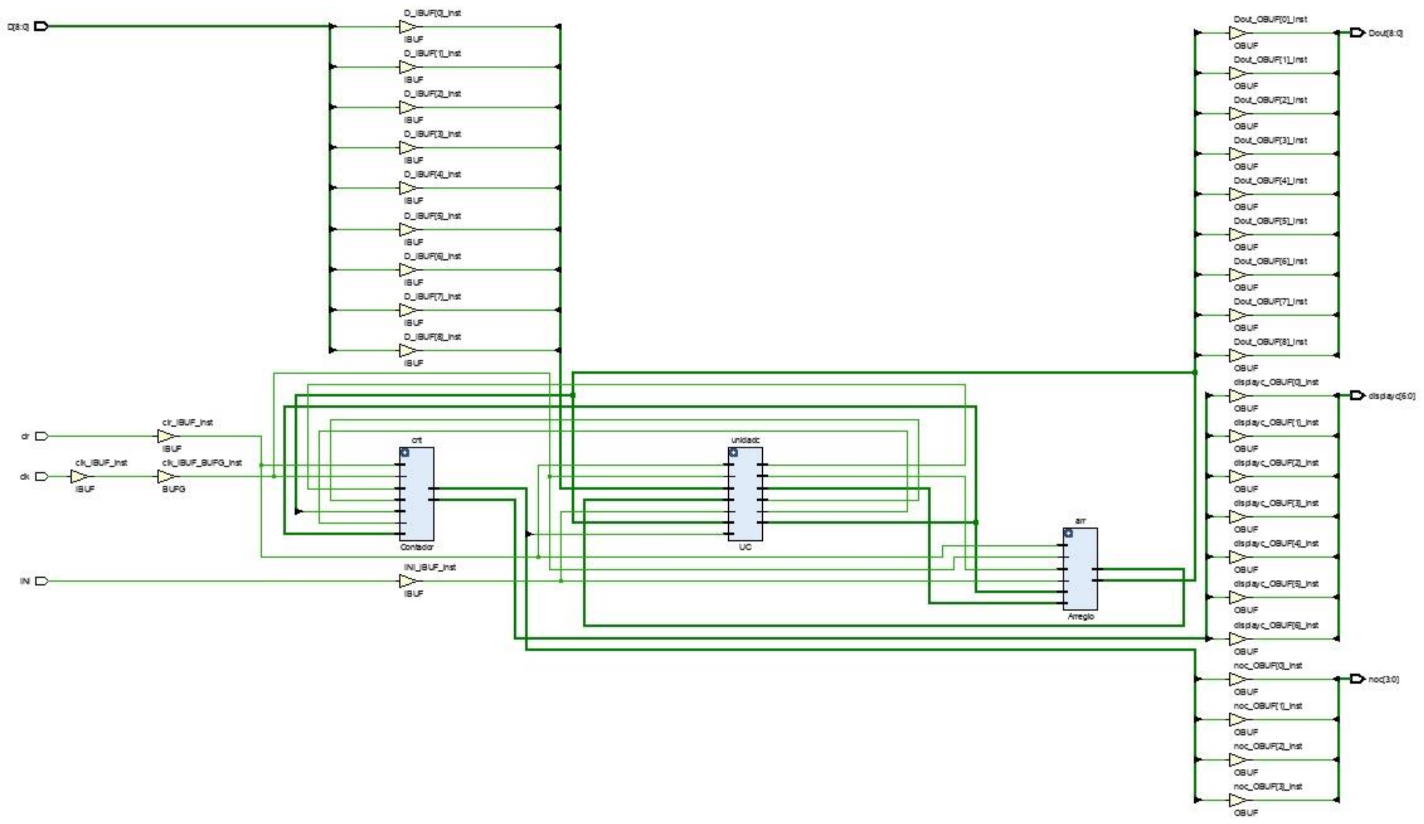

Figura 2.6 Diagrama lógico de la Carta ASM.

Diagrama de Onda de la Unidad de Control.

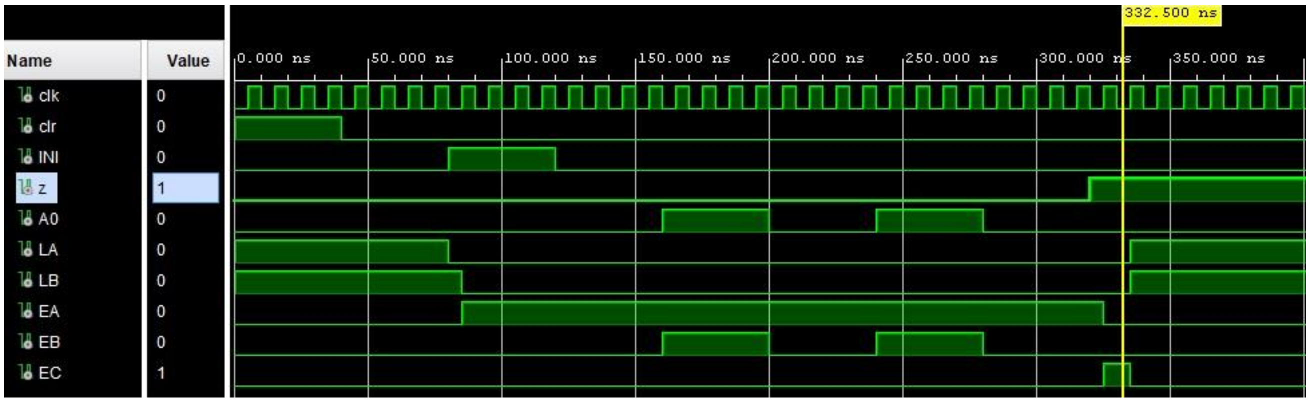

Figura 3. Diagrama de Onda de la Unidad de Control.

Diagrama de Onda del Arreglo.

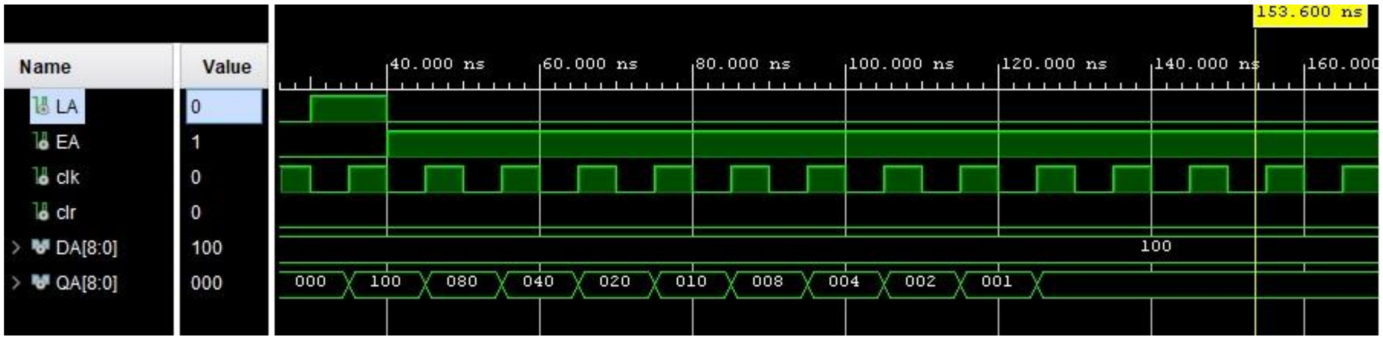

Figura 3.1 Diagrama de Onda del Arreglo.

Diagrama de Onda del Contador.

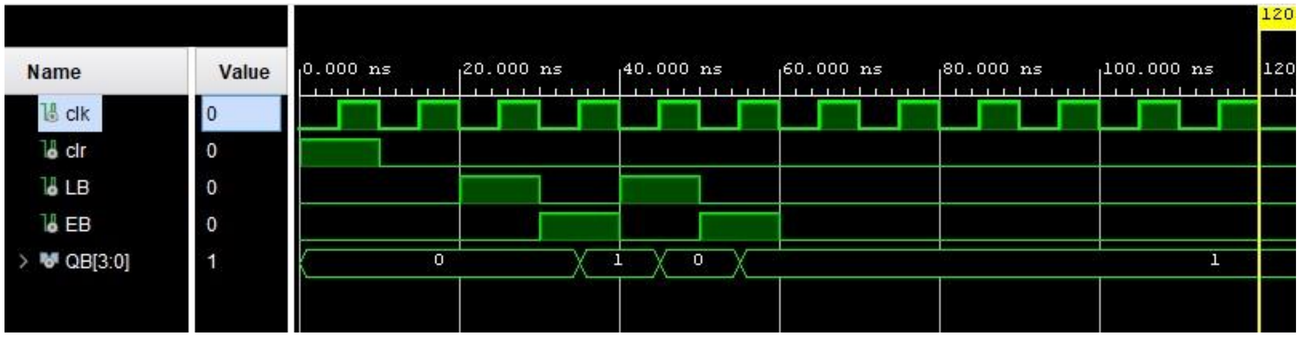

Figura 3.2 Diagrama de Onda del Contador.

Diagrama de Onda de la Carta ASM.

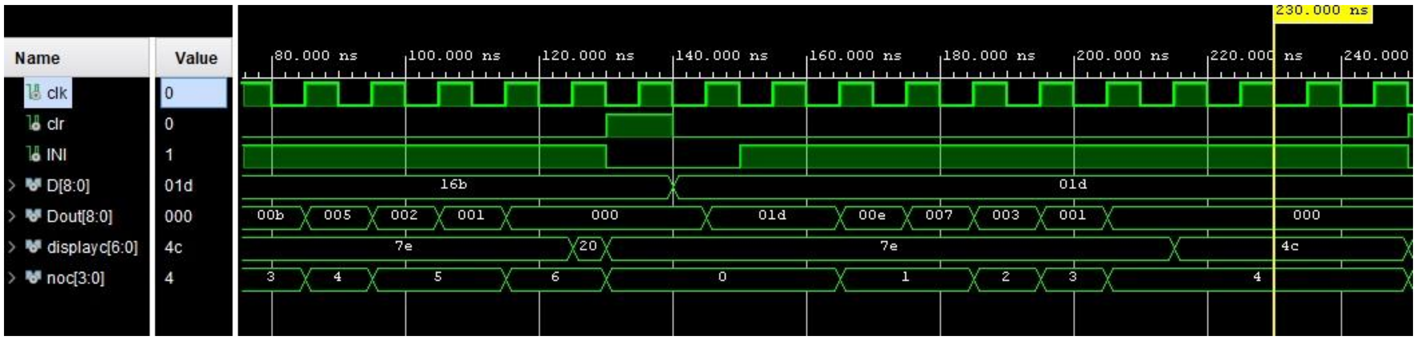

Figura 3.3 Diagrama de Onda de la Carta ASM.

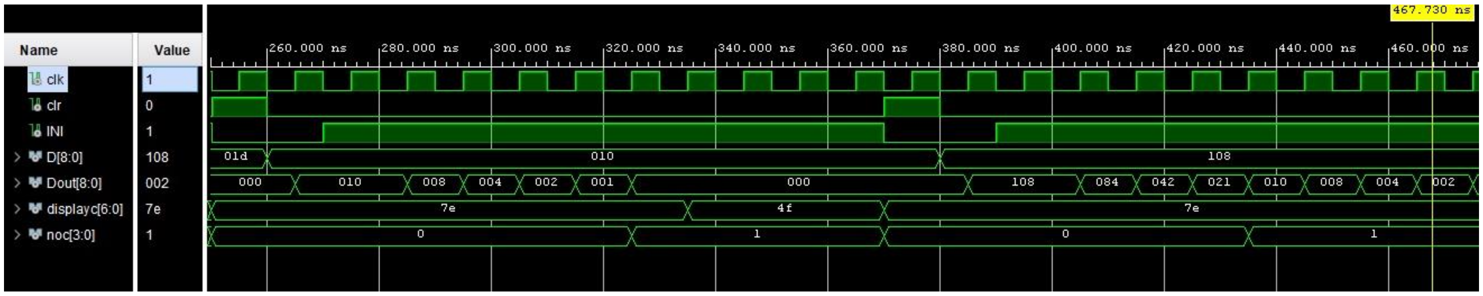

Figura 3.4 Diagrama de Onda de la Carta ASM.

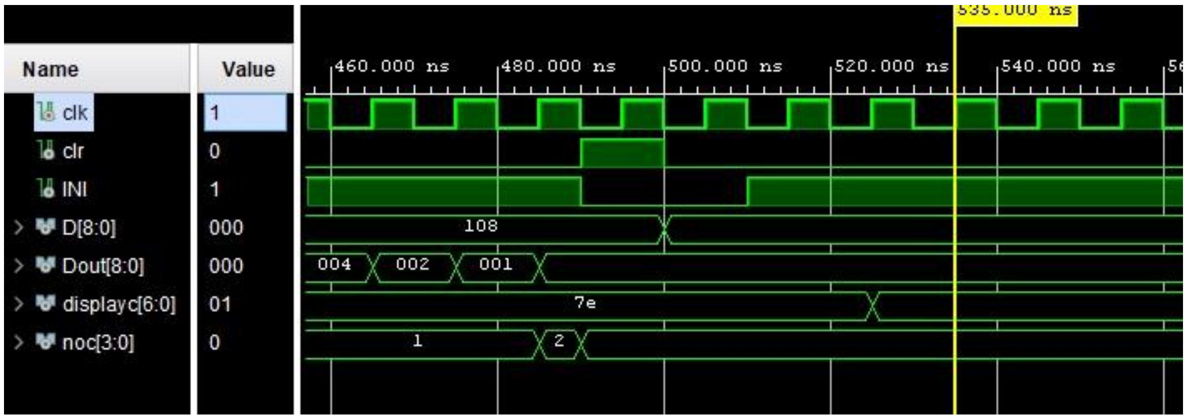

Figura 3.5 Diagrama de Onda de la Carta ASM.