



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.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity UC is
5.     Port(INI, clr, clk, z, A0: in STD_LOGIC;
6.          LA, EA, LB, EB, EC: out STD_LOGIC);
7. end UC;
8.
9. architecture Behavioral of UC is
10.     type edo is (E0, E1, E2);
11.     signal estado_act, estado_sig: edo;
12. begin
13.
14.     process(clr, clk)
15.     begin
16.         if(clr='1') then
17.             estado_act<=E0;
18.         elsif(rising_edge(clk)) then
19.             estado_act<=estado_sig;
20.         end if;
21.     end process;
22.
23.     --aquí se inicia la carta asm
24.     process(estado_act, INI, z, A0)
25.     begin
26.         LA<='0';
27.         EA<='0';
28.         LB<='0';
29.         EB<='0';
30.         EC<='0';
31.
32.         case estado_act is
33.         when E0=>
34.             LB<='1';
35.             if(INI='1') then
36.                 estado_sig<=E1;
37.             else
38.                 LA<='1';
39.                 estado_sig<=E0;
40.             end if;
41.         when E1=>
42.             EA<='1';
43.             if(z='0') then
44.                 if(A0='1') then
45.                     EB<='1';
46.                     estado_sig<=E1;
47.                 else
48.                     estado_sig<=E1;
49.                 end if;
```

```

50.                                     else
51.                                     estado_sig<=E2;
52.                                     end if;
53.                                     when E2=>
54.                                     EC<='1';
55.                                     if(INI='1') then
56.                                     estado_sig<=E2;
57.                                     else
58.                                     estado_sig<=E0;
59.                                     end if;
60.                                     end case;
61.                                     end process;
62.     end Behavioral;

```

Código de Implementación del Arreglo.

```

1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity Arreglo is
5. Port(LA, EA, clk, clr: in STD_LOGIC;
6.       DA: in STD_LOGIC_VECTOR(8 downto 0);
7.       QA: out STD_LOGIC_VECTOR(8 downto 0));
8. end Arreglo;
9.
10. architecture Behavioral of Arreglo is
11.     signal auxQA: STD_LOGIC_VECTOR(8 downto 0);
12. begin
13.     process(LA, EA, clk, clr)
14.     begin
15.         if(clr='1') then
16.             auxQA<=(others=>'0');
17.         elsif(rising_edge(clk)) then
18.             if(LA='0' and EA='0') then
19.                 auxQA<=auxQA;
20.             elsif(LA='1' and EA='0') then
21.                 auxQA<=DA;
22.             elsif(LA='0' and EA='1') then
23.                 auxQA<=to_stdlogicvector(to_bitve
24.                 ctor(auxQA) SRL 1);
25.             end if;
26.         end if;
27.     end process;
28.     QA<=auxQA;
29. end Behavioral;

```

Código de Implementación del Contador.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_ARITH.ALL;
4. use IEEE.STD_LOGIC_UNSIGNED.ALL;
5.
6. entity Contador is
7. Port(LB, EB, clk, clr: in STD_LOGIC;
8.       QB: out STD_LOGIC_VECTOR(3 downto 0));
9. end Contador;
10.
11. architecture Behavioral of Contador is
12. begin
13.     process(clk,clr, LB,EB)
14.     variable auxQB: STD_LOGIC_VECTOR(3 downto 0);
15.     begin
16.         if(clr='1') then
17.             auxQB:=(others=>'0');
18.         elsif(rising_edge(clk)) then
19.             if(LB='0' and EB='0') then
20.                 auxQB:=auxQB;
21.             elsif(LB='1' and EB='0') then
22.                 auxQB:=(others=>'0');
23.             elsif(LB='0' and EB='1') then
24.                 auxQB:=auxQB+1;
25.             end if;
26.         end if;
27.         QB<=auxQB;
28.     end process;
29. end Behavioral;
```

Código de Implementación del Decodificador.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity Decodificador is
5. Port(B: in STD_LOGIC_VECTOR(3 downto 0);
6.       no: out STD_LOGIC_VECTOR(6 downto 0));
7. end Decodificador;
8.
9. architecture Behavioral of Decodificador is
10.     constant n0: STD_LOGIC_VECTOR(6 downto 0):="0000001";
11.     constant n1: STD_LOGIC_VECTOR(6 downto 0):="1001111";
12.     constant n2: STD_LOGIC_VECTOR(6 downto 0):="0010010";
13.     constant n3: STD_LOGIC_VECTOR(6 downto 0):="0000110";
14.     constant n4: STD_LOGIC_VECTOR(6 downto 0):="1001100";
15.     constant n5: STD_LOGIC_VECTOR(6 downto 0):="0100100";
```

```

16.     constant n6: STD_LOGIC_VECTOR(6 downto 0) := "0100000";
17.     constant n7: STD_LOGIC_VECTOR(6 downto 0) := "0001110";
18.     constant n8: STD_LOGIC_VECTOR(6 downto 0) := "0000000";
19.     constant n9: STD_LOGIC_VECTOR(6 downto 0) := "0001100";
20.
21.     begin
22.         process(B)
23.         begin
24.             case B is
25.                 when "0000" => no<=n0;
26.                 when "0001" => no<=n1;
27.                 when "0010" => no<=n2;
28.                 when "0011" => no<=n3;
29.                 when "0100" => no<=n4;
30.                 when "0101" => no<=n5;
31.                 when "0110" => no<=n6;
32.                 when "0111" => no<=n7;
33.                 when "1000" => no<=n8;
34.                 when others => no<=n9;
35.             end case;
36.         end process;
37.     end Behavioral;

```

Código de Implementación del Multiplexor.

```

1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity Multiplexor is
5. Port(nod: in STD_LOGIC_VECTOR(6 downto 0);
6.       EC: in STD_LOGIC;
7.       display: out STD_LOGIC_VECTOR(6 downto 0));
8. end Multiplexor;
9.
10.    architecture Behavioral of Multiplexor is
11.    constant caracter: STD_LOGIC_VECTOR(6 downto 0) := "111111
12.    0";
13.    begin
14.        with EC select
15.            display<=num when '1',
16.            caracter when others;
17.    end Behavioral;

```

Código de Test Bench de la Unidad de Control.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity TbUC is
5. end TbUC;
6.
7. architecture Behavioral of TbUC is
8.
9.     component UC is
10.         Port (INI, clr, clk, z, A0: in STD_LOGIC;
11.             LA, EA, LB, EB,
12.             EC: out STD_LOGIC);
13.     end component;
14.
15.     signal clk, clr INI, z, A0, LA, LB, EA, EB,
16.         EC: STD_LOGIC:='0';
17.
18.     begin
19.
20.         simu : UC
21.             Port map(
22.                 clk=>clk,
23.                 clr=>clr,
24.                 INI=>INI,
25.                 z=>z,
26.                 A0=>A0,
27.                 LA=>LA,
28.                 LB=>LB,
29.                 EA=>EA,
30.                 EB=>EB,
31.                 EC=>EC
32.             );
33.
34.         clock: process begin
35.             clk<='0';
36.             wait for 5 ns;
37.             clk<='1';
38.             wait for 5 ns;
39.         end process;
40.
41.         process
42.         begin
43.             clr<='1';
44.             wait for 40 ns;
45.             clr<='0';
46.             wait for 40 ns;
47.             INI<='1';
48.             wait for 40 ns;
49.             INI<='0';
50.             wait for 40 ns;
```

```

48.         A0<='1';
49.         wait for 40 ns;
50.         A0<='0';
51.         wait for 40 ns;
52.         A0<='1';
53.         wait for 40 ns;
54.         A0<='0';
55.         wait for 40 ns;
56.         z<='1';
57.         wait;
58.     end process;
59. end Behavioral;

```

Código de Test Bench del Arreglo.

```

1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity TbArreglo is
5. end tbArreglo;
6.
7. architecture Behavioral of TbArreglo is
8.
9.     component Arreglo is
10.         Port(LA, EA, clk, clr: in STD_LOGIC;
11.             DA: in STD_LOGIC_VECTOR(8 downto 0);
12.             QA: out STD_LOGIC_VECTOR(8 downto 0));
13.     end component;
14.
15.     signal LA, EA, clk, clr: STD_LOGIC:='0';
16.     signal DA,
17.     QA: STD_LOGIC_VECTOR(8 downto 0):=(others=>'0');
18.
19. begin
20.     arr: Arreglo
21.     Port map(
22.         LA=>LA,
23.         EA=>EA,
24.         clk=>clk,
25.         clr=>clr,
26.         DA=>DA,
27.         QA=>QA
28.     );
29.
30.     reloj: process begin
31.         clk<='0';
32.         wait for 5 ns;
33.         clk<='1';
34.         wait for 5 ns;

```

```

34.         end process;
35.
36.         process
37.         begin
38.             clr<='1';
39.             wait for 10 ns;
40.             clr<='0';
41.             DA<='100000000';
42.             wait for 10 ns;
43.             LA<='0';
44.             EA<='0';
45.             wait for 10 ns;
46.             LA<='1';
47.             EA<='0';
48.             wait for 10 ns;
49.             LA<='0';
50.             EA<='1';
51.             wait;
52.         end process;
53.     end Behavioral;

```

Código de Test Bench del Contador.

```

1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_ARITH.ALL;
4. use IEEE.STD_LOGIC_UNSIGNED.ALL;
5.
6. entity TbContador is
7. end TbContador;
8.
9. architecture Behavioral of TbContador is
10.
11.     component Contador is
12.     Port(LB, EB, clk, clr: in STD_LOGIC;
13.         QB: out STD_LOGIC_VECTOR(3 downto 0));
14.     end component;
15.
16.     signal clk,clr,LB,EB: STD_LOGIC:='0';
17.     signal QB: STD_LOGIC_VECTOR(3 downto 0):=(others
=>'0');
18.     begin
19.         cnt: Contador
20.         Port map(
21.             clk=>clk,
22.             clr=>clr,
23.             LB=>LB,
24.             EB=>EB,
25.             QB=>QB

```



```

26.         );
27.
28.         clock: process begin
29.             clk<='0';
30.             wait for 5 ns;
31.             clk<='1';
32.             wait for 5 ns;
33.         end process;
34.
35.         process
36.         begin
37.             clr<='1';
38.             wait for 10 ns;
39.             clr<='0';
40.             wait for 10 ns;
41.             LB<='1';
42.             EB<='0';
43.             wait for 10 ns;
44.             LB<='0';
45.             EB<='1';
46.             wait for 10 ns;
47.             LB<='1';
48.             EB<='0';
49.             wait for 10 ns;
50.             LB<='0';
51.             EB<='1';
52.             wait for 10 ns;
53.             LB<='0';
54.             EB<='0';
55.             wait;
56.         end process;
57.     end Behavioral;

```

Código de paquete de la Carta ASM.

```
1. library IEEE;
2. use IEEE.std_logic_1164.All;
3.
4. package PaqueteASM is
5.
6.     component UC is
7.         Port(INI, clr, clk, z, A0: in STD_LOGIC;
8.             LA, EA, LB, EB, EC: out STD_LOGIC);
9.     end component;
10.
11.     component Arreglo is
12.         Port(LA, EA, clk, clr: in STD_LOGIC;
13.             DA: in STD_LOGIC_VECTOR(8 downto 0);
14.             QA: out STD_LOGIC_VECTOR(8 downto 0));
15.     end component;
16.
17.     component Contador is
18.         Port(LB, EB, clk, clr: in STD_LOGIC;
19.             QB: out STD_LOGIC_VECTOR(3 downto 0));
20.     end component;
21.
22.     component Multiplexor is
23.         Port(nod: in STD_LOGIC_VECTOR(6 downto 0);
24.             EC: in STD_LOGIC;
25.             display: out STD_LOGIC_VECTOR(6 downto 0
26.         ));
27.     end component;
28.
29.     component Decodificador is
30.         Port(B: in STD_LOGIC_VECTOR(3 downto 0);
31.             no: out STD_LOGIC_VECTOR(6 downto 0));
32.     end component;
33. end package;
```

Código de Implementación de la Carta ASM.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use work.PaqueteASM.all;
4.
5. entity CartaASM is
6. Port (clk, clr, INI: in STD_LOGIC;
7.       D: in STD_LOGIC_VECTOR(8 downto 0);
8.       Dout: out STD_LOGIC_VECTOR(8 downto 0);
9.       noc: out STD_LOGIC_VECTOR(3 downto 0);
10.      displayc: out STD_LOGIC_VECTOR(6 downto 0));
11. end CartaASM;
12.
13. architecture Behavioral of CartaASM is
14.
15.
16.      signal auxZ, auxLA, auxLB, auxEA, auxEB,
17.      auxEC: STD_LOGIC:='0';
18.      signal auxA: STD_LOGIC_VECTOR(8 downto 0):=(others=>'0');
19.      signal auxB: STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
20.      signal auxNo: STD_LOGIC_VECTOR(6 downto 0):=(others=>'0');
21.
22. begin
23.     arr: Arreglo
24.     Port map(
25.     LA=>auxLA,
26.     EA=>auxEA,
27.     clk=>clk,
28.     clr=>clr,
29.     DA=>D,
30.     QA=>auxA
31.     );
32.
33.     auxZ<='1' when auxA="00000000" else '0';
34.
35.     unidadc: UC
36.     Port map(
37.     clk=>clk,
38.     clr=>clr,
39.     INI=>INI,
40.     z=>auxZ,
41.     A0=>auxA(0),
42.     LA=>auxLA,
43.     LB=>auxLB,
44.     EA=>auxEA,
45.     EB=>auxEB,
46.     EC=>auxEC
```

```

46.         );
47.
48.         cnt: Contador
49.         Port map(
50.             clk=>clk,
51.             clr=>clr,
52.             LB=>auxLB,
53.             EB=>auxEB,
54.             QB=>auxB
55.         );
56.
57.         dec: Decodificador
58.         Port map(
59.             B=>auxB,
60.             no=>auxNo
61.         );
62.
63.         mux: Multiplexor
64.         Port map(
65.             nod=>auxNo,
66.             EC=>auxEC,
67.             display=>displayc
68.         );
69.
70.         noc<=auxB;
71.         Dout<=auxA;
72.     end Behavioral;

```

Código de Test Bench de la Carta ASM.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use work.PaqueteASM.all;
4.
5. entity TbCartaASM is
6. end TbCartaASM;
7.
8. architecture Behavioral of TbCartaASM is
9.
10.     component CartaASM is
11.     Port(clk, clr, INI: in STD_LOGIC;
12.         D: in STD_LOGIC_VECTOR(8 downto 0);
13.         Dout: out STD_LOGIC_VECTOR(8 downto 0);
14.         noc: out STD_LOGIC_VECTOR(3 downto 0);
15.         displayc: out STD_LOGIC_VECTOR(6 downto
16.         0));
17.
18.     end component;
19.
20.     signal clk,clr,INI: STD_LOGIC;
21.     signal D, Dout: STD_LOGIC_VECTOR(8 downto 0);
22.     signal displayc: STD_LOGIC_VECTOR(6 downto 0);
23.     signal noc: STD_LOGIC_VECTOR(3 downto 0);
24.
25. begin
26.
27.     CASM: CartaASM
28.     Port map(
29.         clk=>clk,
30.         clr=>clr,
31.         INI=>INI,
32.         D=>D,
33.         Dout=>Dout,
34.         noc=>noc,
35.         displayc=>displayc
36.     );
37.
38.     clock: process begin
39.         clk<='0';
40.         wait for 5 ns;
41.         clk<='1';
42.         wait for 5 ns;
43.         end process;
44.
45.     process
46.     begin
47.         INI<='0';
48.         clr<='1';
49.         wait for 10 ns;
```

```

49.         clr<='0';
50.         D<="101101011";
51.         wait for 10 ns;
52.
53.         INI<='1';
54.         wait for 110 ns;
55.
56.         INI<='0';
57.         clr<='1';
58.         wait for 10 ns;
59.
60.         clr<='0';
61.         D<="000011101";
62.         wait for 10 ns;
63.
64.         INI<='1';
65.         wait for 100 ns;
66.
67.         INI<='0';
68.         clr<='1';
69.         wait for 10 ns;
70.
71.         clr<='0';
72.         D<="000010000";
73.         wait for 10 ns;
74.
75.         INI<='1';
76.         wait for 100 ns;
77.
78.         INI<='0';
79.         clr<='1';
80.         wait for 10 ns;
81.
82.         clr<='0';
83.         D<="100001000";
84.         wait for 10 ns;
85.
86.         INI<='1';
87.         wait for 100 ns;
88.
89.         INI<='0';
90.         clr<='1';
91.         wait for 10 ns;
92.
93.         clr<='0';
94.         D<="000000000";
95.         wait for 10 ns;
96.
97.         INI<='1';
98.         wait;
99.     end process;
100. end Behavioral;

```

Diagrama RTL de la Unidad de Control.

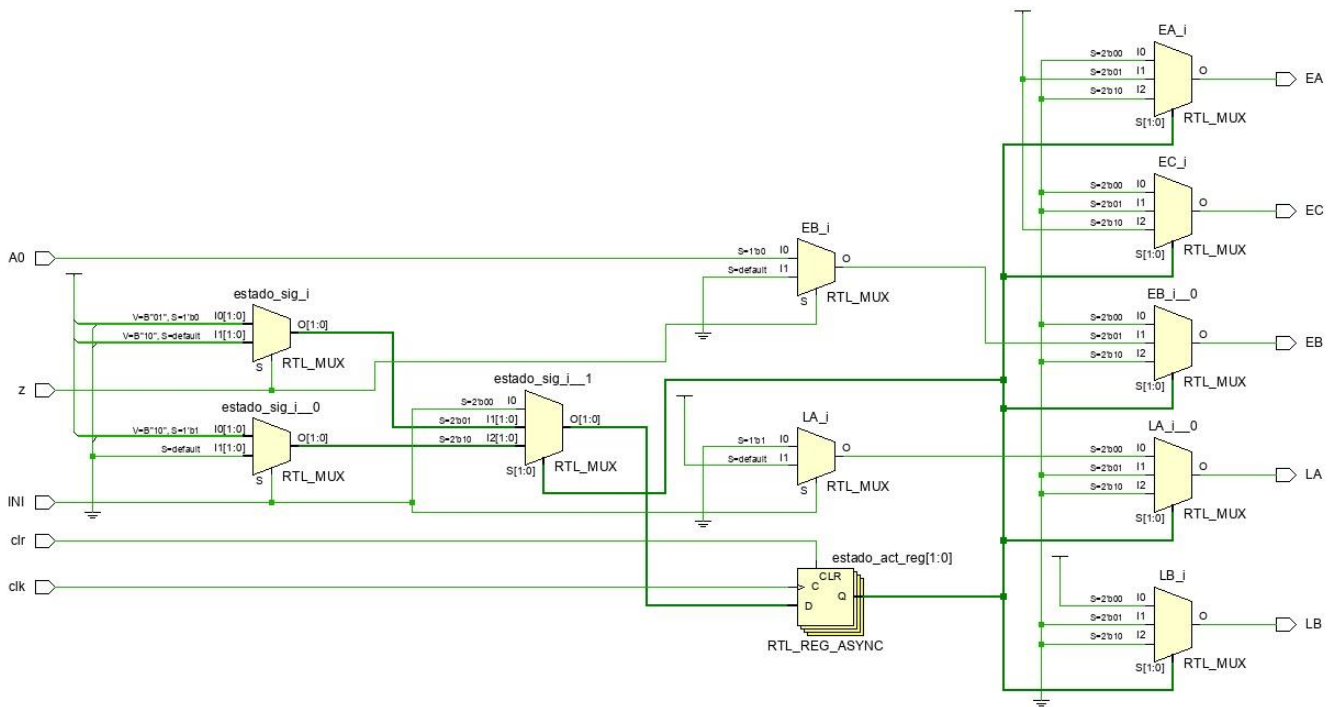


Figura 1. Diagrama RTL de la Unidad de Control.

Diagrama RTL de Arreglo.

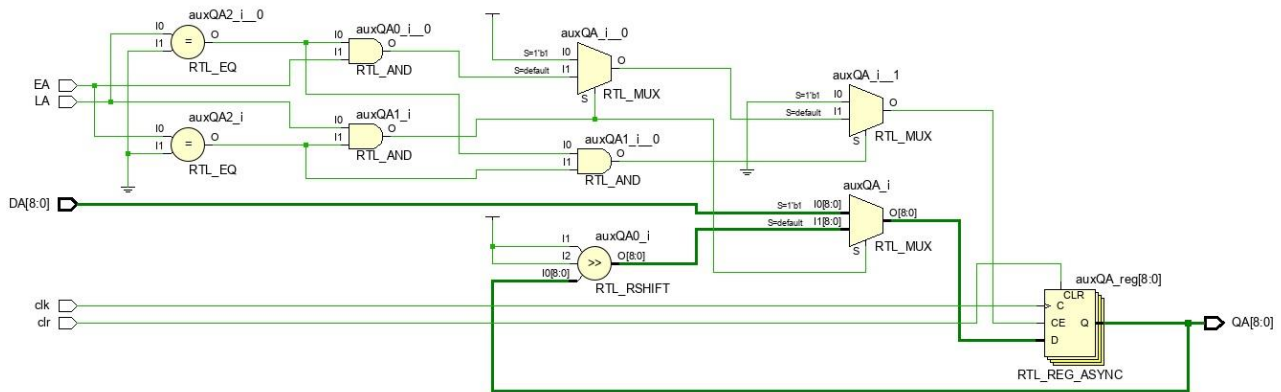


Figura 1.1 Diagrama RTL del Arreglo.

Diagrama RTL de Contador.

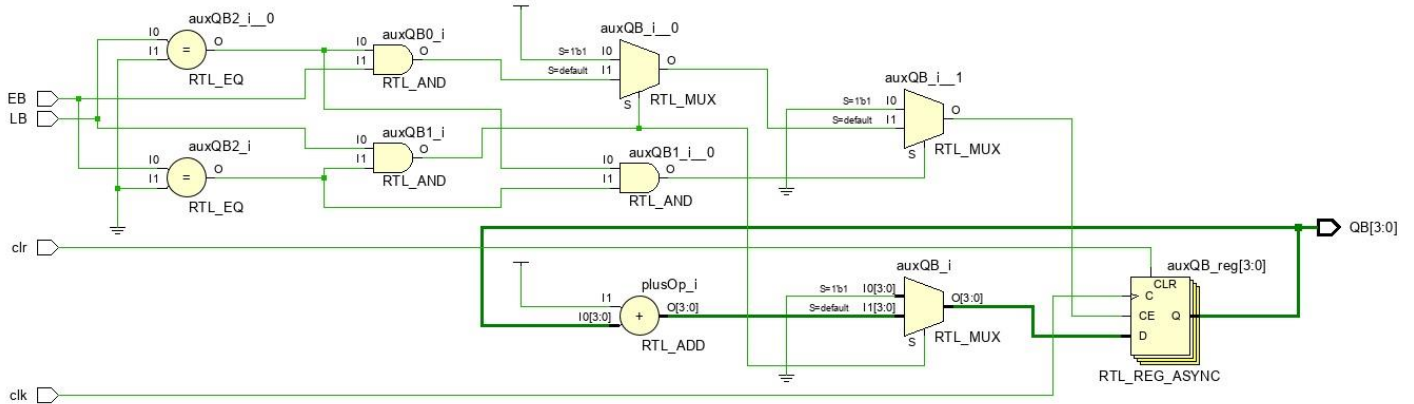


Figura 1.2 Diagrama RTL del Contador.

Diagrama RTL del Decodificador.

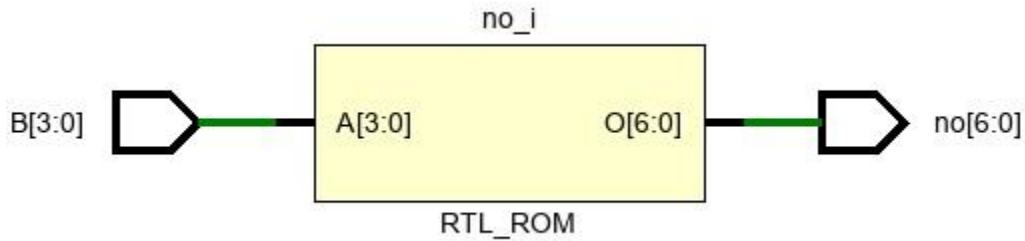


Figura 1.3 Diagrama RTL del Decodificador.

Diagrama RTL del Multiplexor.

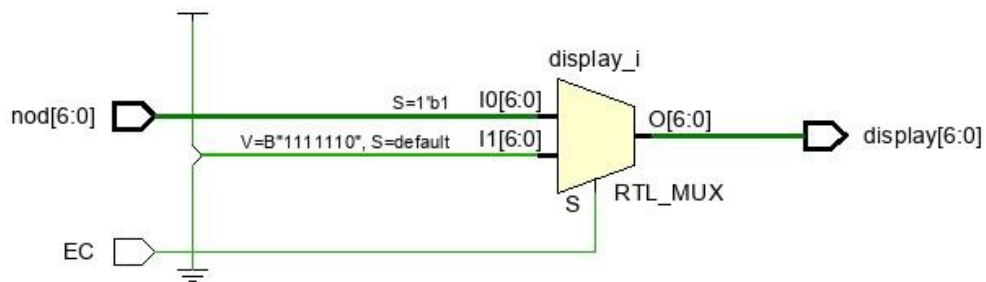


Figura 1.4 Diagrama RTL del Multiplexor.

Diagrama RTL de Carta ASM.

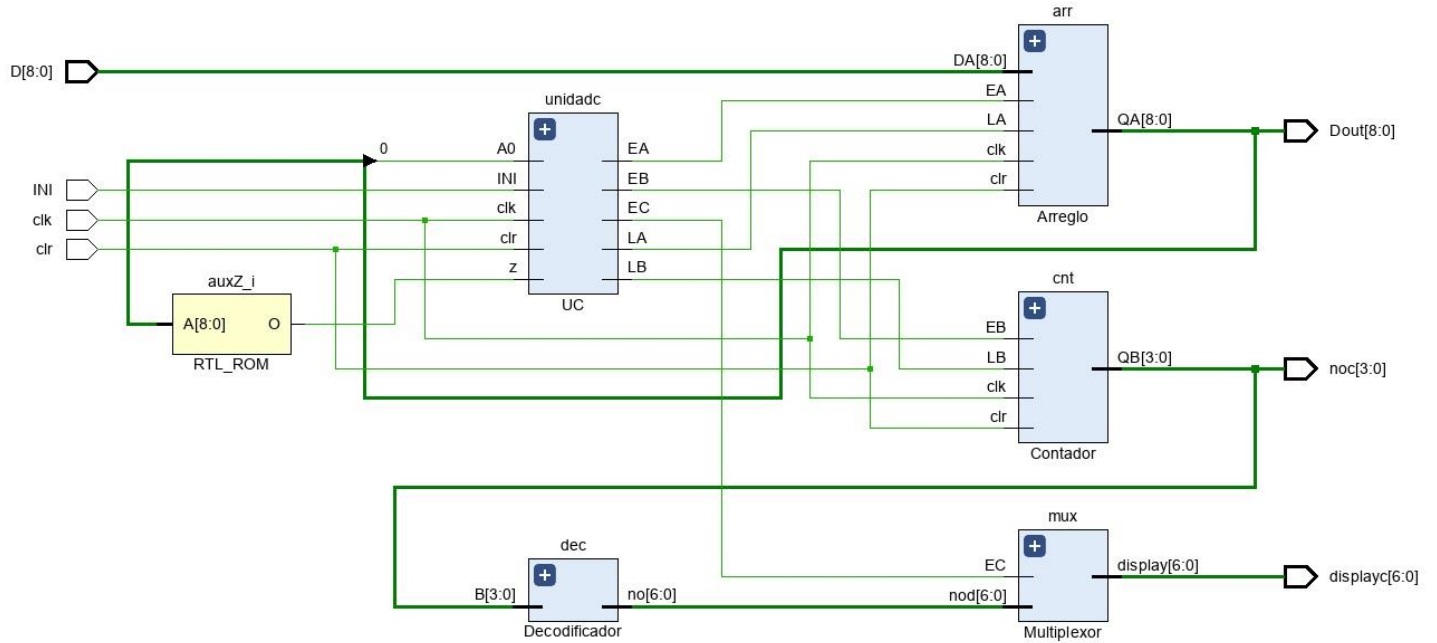


Figura 1.5 Diagrama RTL de la Carta ASM

Diagrama lógico de Unidad de Control.

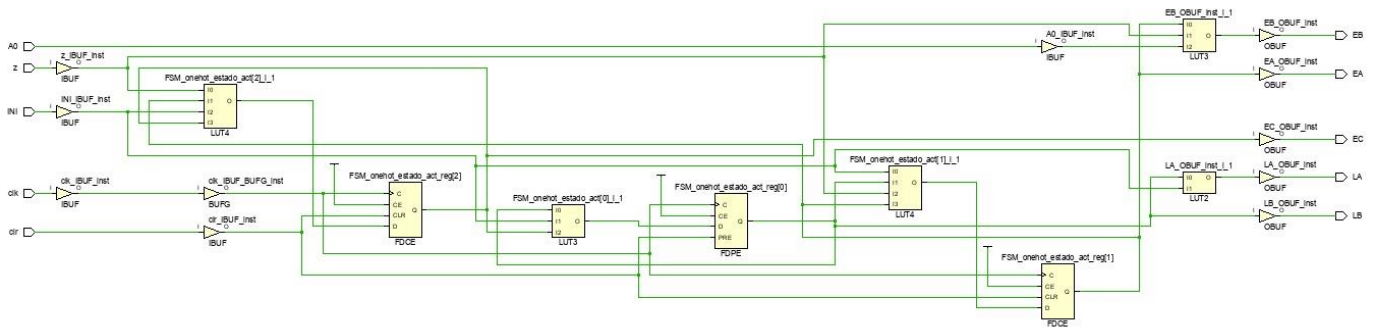


Figura 2 Diagrama lógico de la Unidad de Control.

Diagrama lógico del Arreglo.

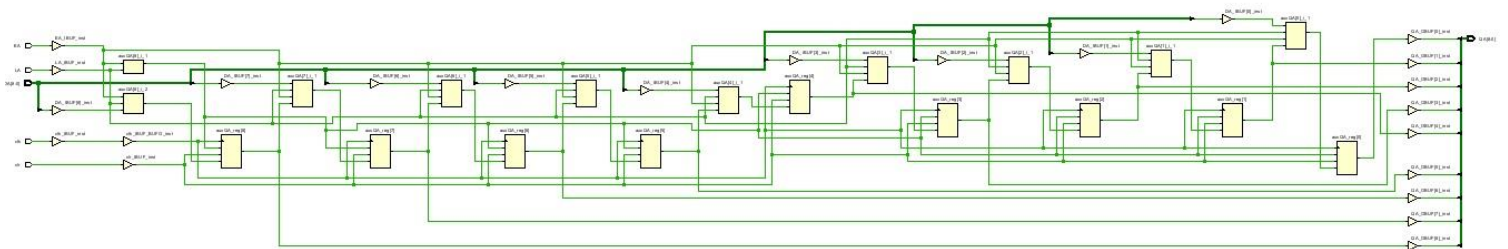


Figura 2.1 Diagrama lógico del Arreglo.

Diagrama lógico del Contador.

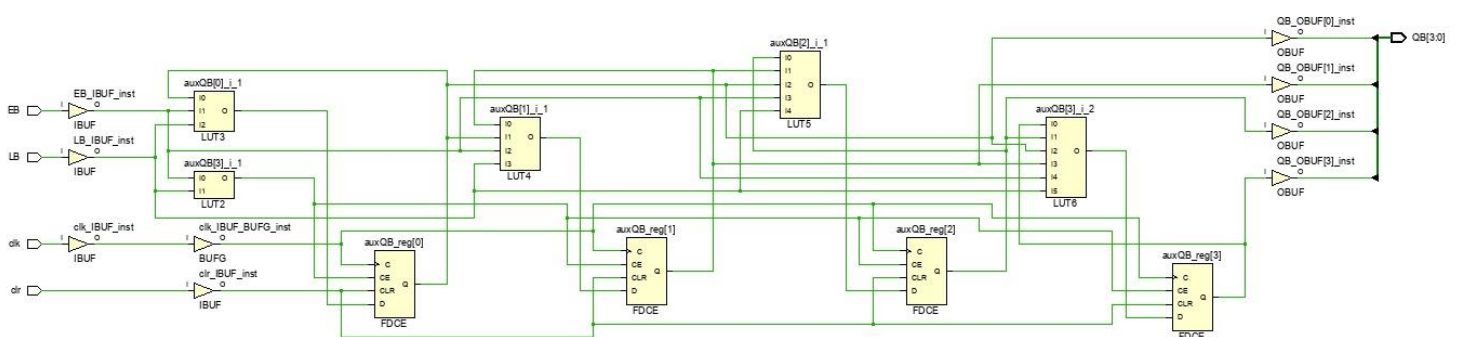


Figura 2.3 Diagrama lógico del Contador.

Diagrama lógico del Decodificador.

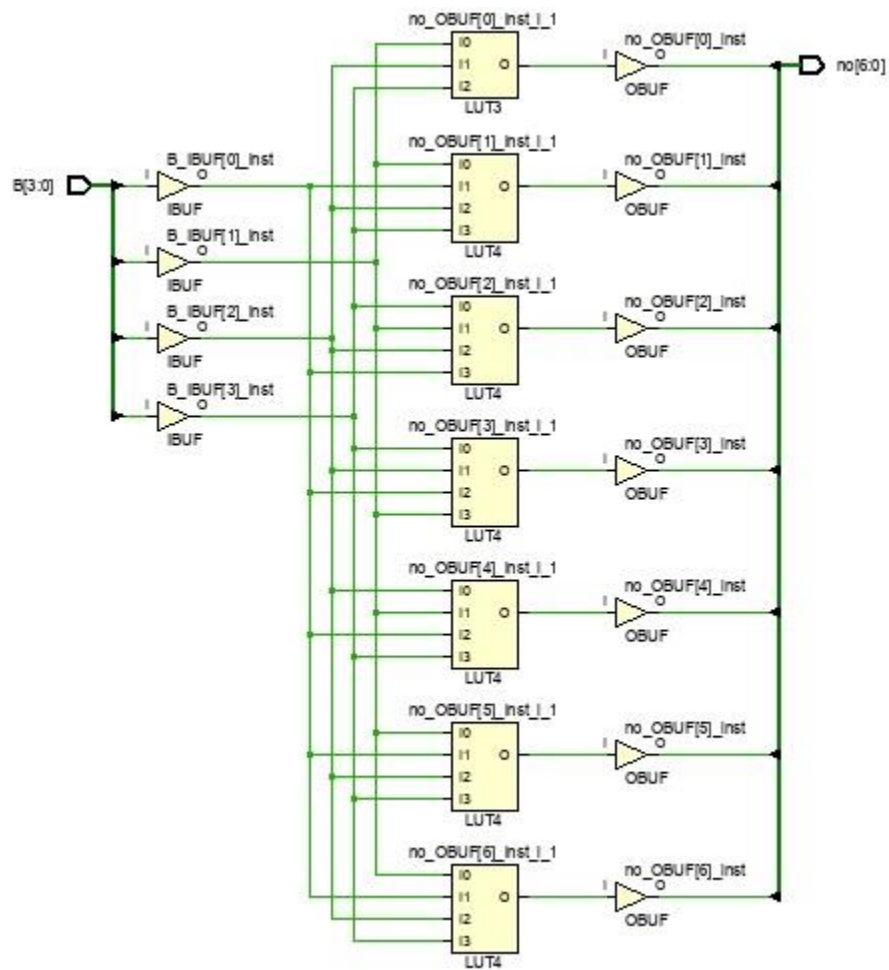


Figura 2.4 Diagrama lógico del Decodificador.

Diagrama lógico del Multiplexor.

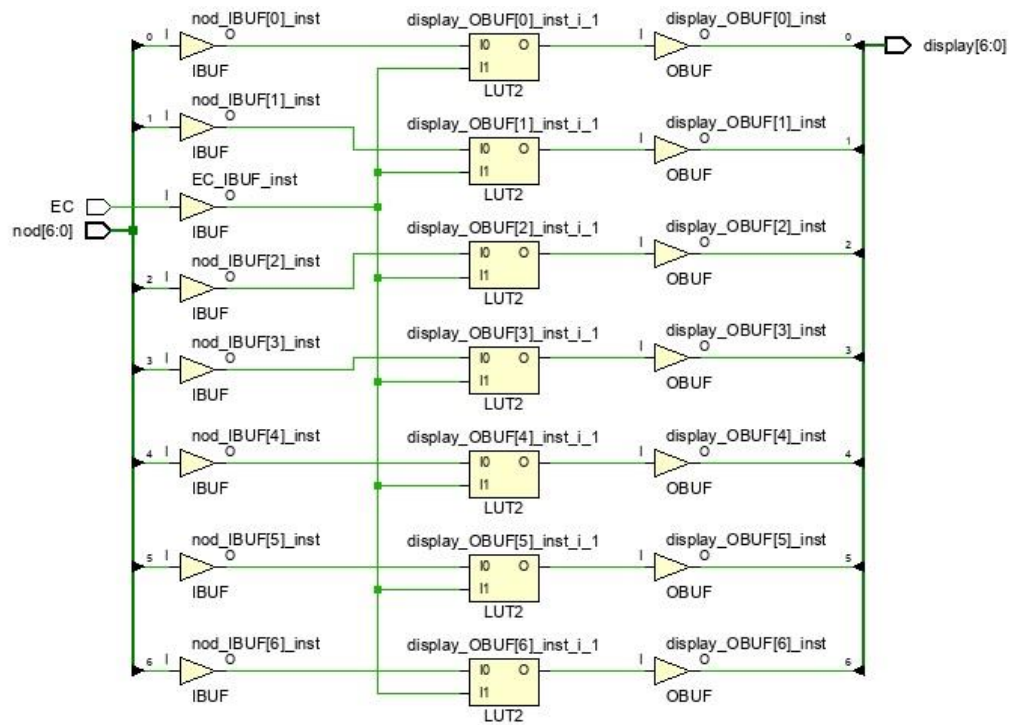


Figura 2.5 Diagrama lógico del Multiplexor.

Diagrama lógico de la Carta ASM.

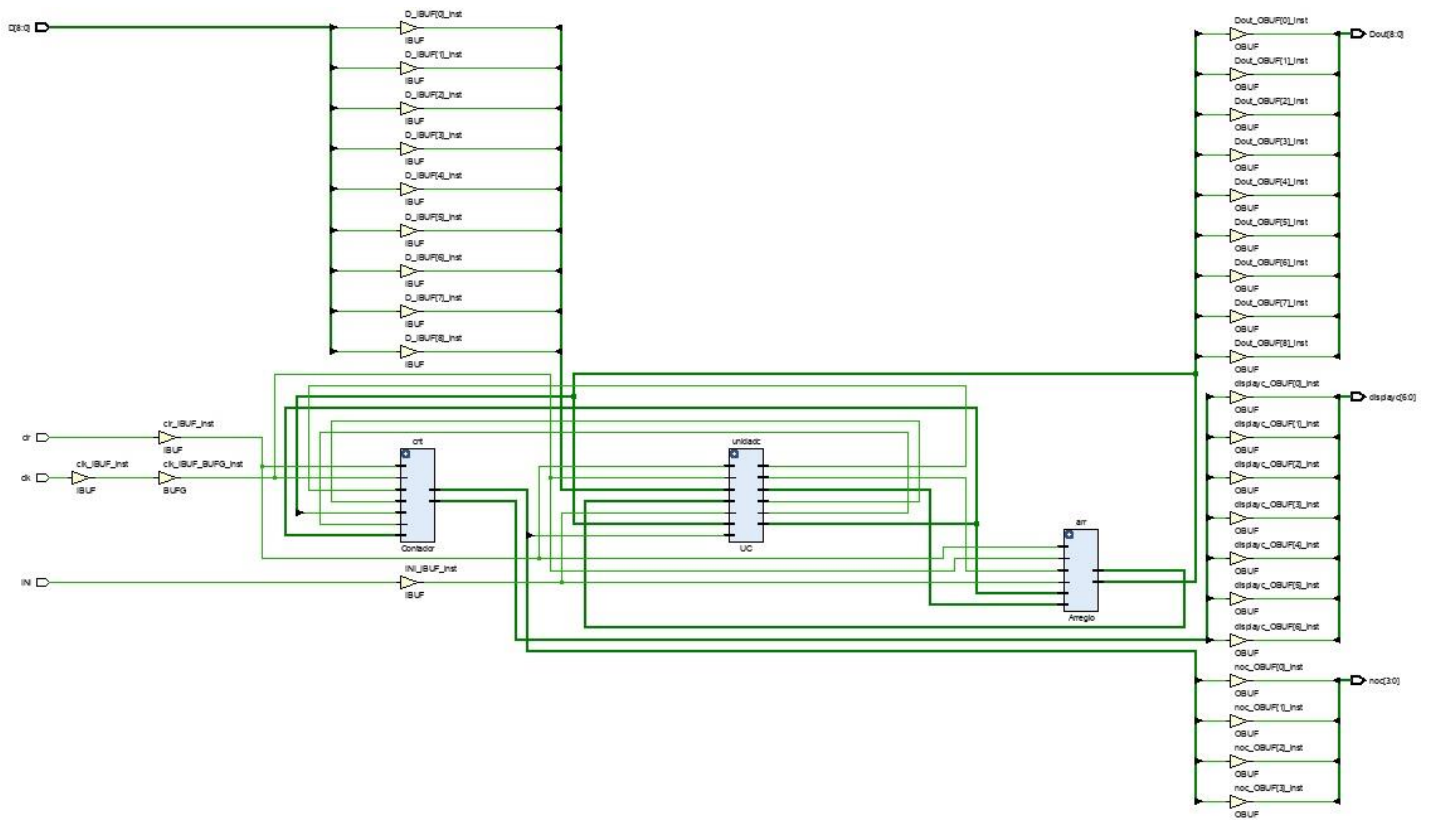


Figura 2.6 Diagrama lógico de la Carta ASM.

Diagrama de Onda de la Unidad de Control.



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

Diagrama de Onda del Arreglo.

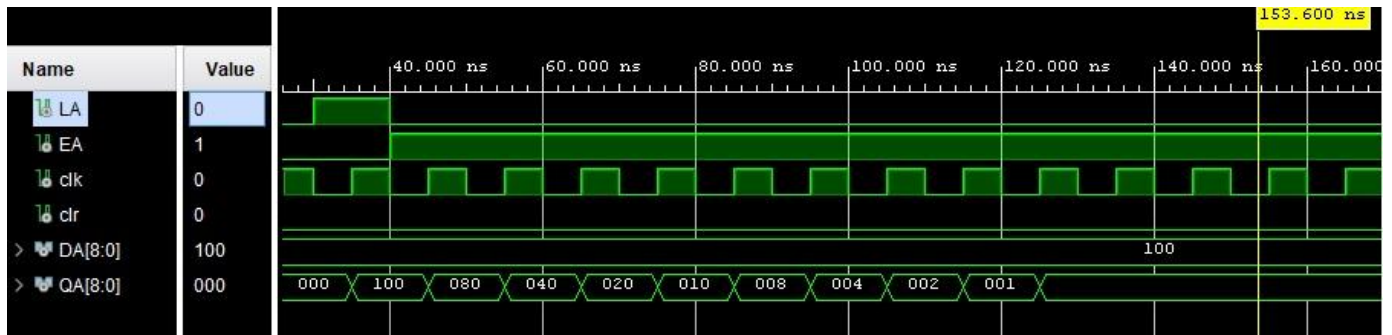


Figura 3.1 Diagrama de Onda del Arreglo.

Diagrama de Onda del Contador.

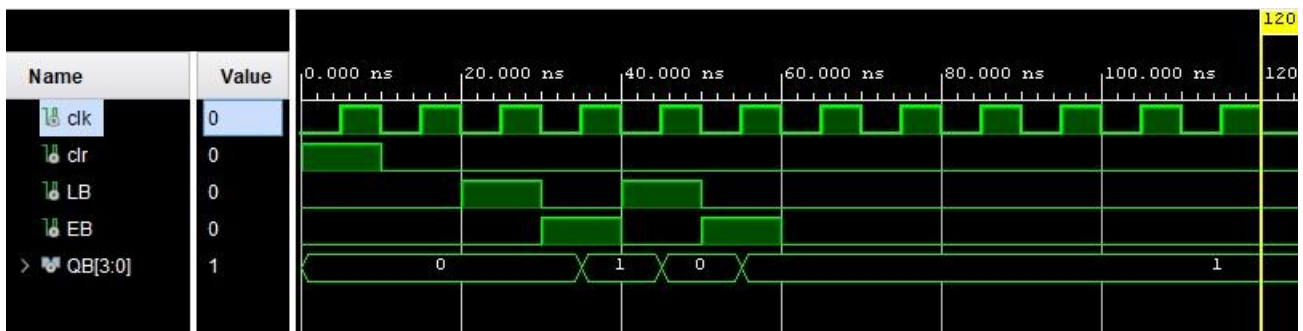


Figura 3.2 Diagrama de Onda del Contador.

Diagrama de Onda de la Carta ASM.

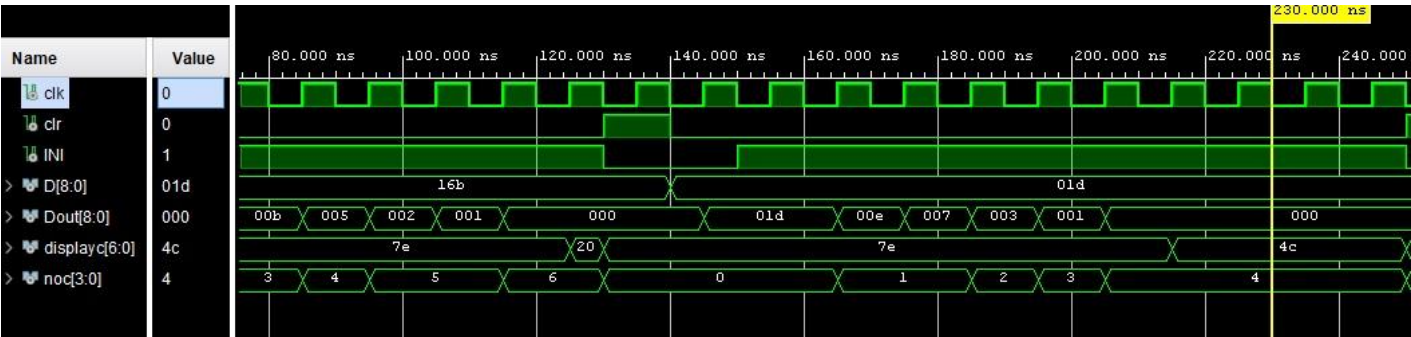


Figura 3.3 Diagrama de Onda de la Carta ASM.

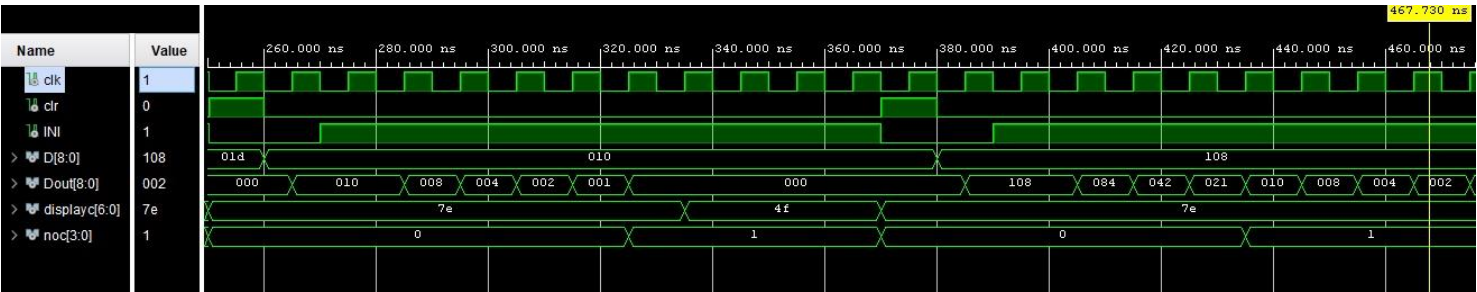


Figura 3.4 Diagrama de Onda de la Carta ASM.

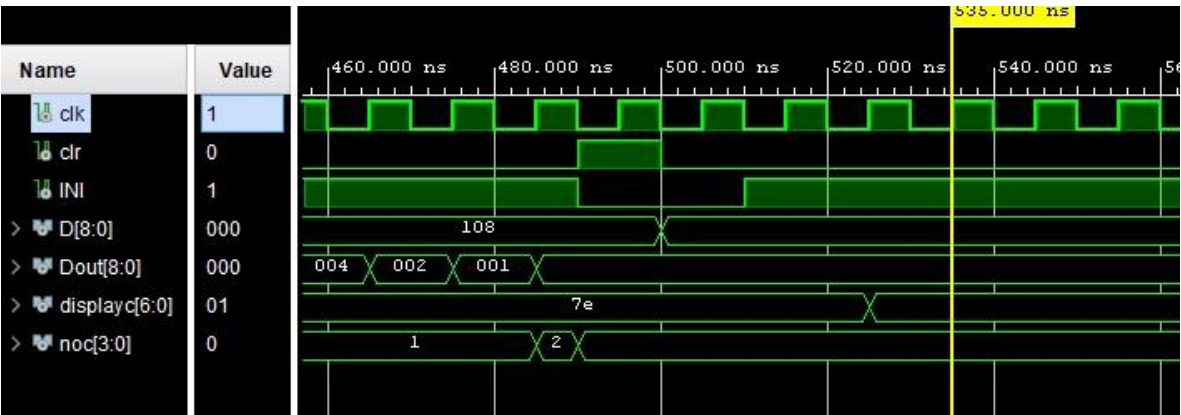


Figura 3.5 Diagrama de Onda de la Carta ASM.