Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 12 - Cartas ASM

Unidad de aprendizaje: Arquitectura de Computadoras

Grupo: 3CV1

Alumno(a):
Ramos Diaz Enrique

Profesor(a):
Vega García Nayeli

28 de mayo 2020

Índice

1. Código de implementación

1.1. Unidad de Control

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

entity UnidadControl is
    Port ( clk, clr, ini, z, a0 : in STD_LOGIC;
           LA, LB, EA, EB, EC : out STD_LOGIC);
end UnidadControl;

architecture Behavioral of UnidadControl is
    type estados is (e0, e1, e2);
    signal edo_act, edo_sig : estados;
begin
    --Control de estados
    process(clr, clk)
    begin
        if(clr = '1') then
            edo_act <= e0;
        elsif (rising_edge(clk)) then
            edo_act <= edo_sig;
        end if;
    end process;

    --Carta ASM
    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 = '1') then
                    edo_sig <= e1;
                else
                    LA <= '1';
                    edo_sig <= e0;
                end if;
            when e1 =>
```

```
                EA <= '1';
                if (z = '0') then
                    if (a0 = '1') then
                        EB <= '1';
                        edo_sig <= e1;
                    else
                        edo_sig <= e1;
                    end if;
                else
                    edo_sig <= e2;
                end if;
            when e2 =>
                EC <= '1';
                if (ini = '1') then
                    edo_sig <= e2;
                else
                    edo_sig <= e0;
                end if;
        end case;
    end process;
end Behavioral;
```

1.2. Registro

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

entity Registro is
    Port ( LA, EA, clk, clr : in STD_LOGIC;
           dato : in STD_LOGIC_VECTOR (8 downto 0);
           A : out STD_LOGIC_VECTOR (8 downto 0));
end Registro;

architecture Behavioral of Registro is
    signal auxA: STD_LOGIC_VECTOR (8 downto 0);
begin
    process(LA, EA, clk, clr)
    begin
        if (clr = '1') then
            auxA <= (others => '0');
        elsif (rising_edge(clk)) then
            if (LA = '0' and EA = '0') then
                auxA <= auxA;
```

```
            elsif (LA = '1' and EA = '0') then
                auxA <= dato;
            elsif (LA = '0' and EA = '1') then
                auxA <= to_stdlogicvector(to_bitvector(auxA) SRL 1);
            end if;
        end if;
    end process;
    A <= auxA;
end Behavioral;
```

1.3. 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 ( clk, clr, LB, EB : in STD_LOGIC;
           B : out STD_LOGIC_VECTOR (3 downto 0));
end Contador;

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

1.4. Decodificador

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

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

architecture Behavioral of Decodificador is
    constant dig0: STD_LOGIC_VECTOR (6 downto 0):= "0000001";
    constant dig1: STD_LOGIC_VECTOR (6 downto 0):= "1001111";
    constant dig2: STD_LOGIC_VECTOR (6 downto 0):= "0010010";
    constant dig3: STD_LOGIC_VECTOR (6 downto 0):= "0000110";
    constant dig4: STD_LOGIC_VECTOR (6 downto 0):= "1001100";
    constant dig5: STD_LOGIC_VECTOR (6 downto 0):= "0100100";
    constant dig6: STD_LOGIC_VECTOR (6 downto 0):= "0100000";
    constant dig7: STD_LOGIC_VECTOR (6 downto 0):= "0001110";
    constant dig8: STD_LOGIC_VECTOR (6 downto 0):= "0000000";
    constant dig9: STD_LOGIC_VECTOR (6 downto 0):= "0001100";
begin
    process(B)
    begin
        case B is
            when "0000" => num <= dig0;
            when "0001" => num <= dig1;
            when "0010" => num <= dig2;
            when "0011" => num <= dig3;
            when "0100" => num <= dig4;
            when "0101" => num <= dig5;
            when "0110" => num <= dig6;
            when "0111" => num <= dig7;
            when "1000" => num <= dig8;
            when others => num <= dig9;
        end case;
    end process;
end Behavioral;
```

1.5. Multiplexor

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

entity Multiplexor is
    Port ( num : 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 <= num when '1',
        guion when others;
end Behavioral;
```

1.6. Cartas ASM

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

entity CartasASM is
    Port ( clk, clr, ini : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (8 downto 0);
           Dout : out STD_LOGIC_VECTOR (8 downto 0);
           num : out STD_LOGIC_VECTOR (3 downto 0);
           display : out STD_LOGIC_VECTOR (6 downto 0));
end CartasASM;

architecture Behavioral of CartasASM is
    component UnidadControl is
        Port ( clk, clr, ini, z, a0 : in STD_LOGIC;
               LA, LB, EA, EB, EC : out STD_LOGIC);
    end component;

    component Registro is
        Port ( LA, EA, clk, clr : in STD_LOGIC;
               dato : in STD_LOGIC_VECTOR (8 downto 0);
               A : out STD_LOGIC_VECTOR (8 downto 0));
    end component;
```

```
23
24      component Contador is
25          Port ( clk, clr, LB, EB : in STD_LOGIC;
26                 B : out STD_LOGIC_VECTOR (3 downto 0));
27      end component;
28
29      component Decodificador is
30          Port ( B : in STD_LOGIC_VECTOR (3 downto 0);
31                 num : out STD_LOGIC_VECTOR (6 downto 0));
32      end component;
33
34      component Multiplexor is
35          Port ( num : in STD_LOGIC_VECTOR (6 downto 0);
36                 ec : in STD_LOGIC;
37                 display : out STD_LOGIC_VECTOR (6 downto 0));
38      end component;
39
40      signal auxZ, auxLA, auxLB, auxEA, auxEB, auxEC : STD_LOGIC := '0';
41      signal auxA : STD_LOGIC_VECTOR (8 downto 0) := (others => '0');
42      signal auxB : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');
43      signal auxNum : STD_LOGIC_VECTOR (6 downto 0) := (others => '0');
44  begin
45
46      reg : Registro Port map (
47          LA => auxLA,
48          EA => auxEA,
49          clk => clk,
50          clr => clr,
51          dato => D,
52          A => auxA
53      );
54
55      auxZ <= '1' when auxA = "000000000" else '0';
56
57      uc : UnidadControl Port map (
58          clk => clk,
59          clr => clr,
60          ini => ini,
61          z => auxZ,
62          a0 => auxA(0),
63          LA => auxLA,
64          LB => auxLB,
65          EA => auxEA,
66          EB => auxEB,
```

```
        EC => auxEC
    );

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

    deco : Decodificador Port map (
        B => auxB,
        num => auxNum
    );

    mux : Multiplexor Port map (
        num => auxNum,
        EC => auxEC,
        display => display
    );

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

2. Código de simulación

2.1. Unidad de Control

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

entity tbUnidadControl is
end tbUnidadControl;

architecture Behavioral of tbUnidadControl is
    component UnidadControl is
        Port ( clk, clr, ini, z, a0 : in STD_LOGIC;
               LA, LB, EA, EB, EC : out STD_LOGIC);
    end component;

    signal clk, clr, ini, z, a0, LA, LB, EA, EB, EC : STD_LOGIC := '0';
```

```
14 begin
15     ASM : UnidadControl Port map (
16         clk => clk,
17         clr => clr,
18         ini => ini,
19         z => z,
20         a0 => a0,
21         LA => LA,
22         LB => LB,
23         EA => EA,
24         EB => EB,
25         EC => EC
26     );
27 
28     reloj : 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 30 ns;
39         clr <= '0';
40         wait for 60 ns;
41         ini <= '1';
42         wait for 10 ns;
43         ini <= '0';
44         wait for 50 ns;
45         a0 <= '1';
46         wait for 10 ns;
47         a0 <= '0';
48         wait for 20 ns;
49         a0 <= '1';
50         wait for 10 ns;
51         a0 <= '0';
52         wait for 120 ns;
53         z <= '1';
54         wait;
55     end process;
56 end Behavioral;
```

2.2. Registro

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

entity tbRegistro is
end tbRegistro;

architecture Behavioral of tbRegistro is
    component Registro is
        Port ( LA, EA, clk, clr : in STD_LOGIC;
               dato : in STD_LOGIC_VECTOR (8 downto 0);
               A : out STD_LOGIC_VECTOR (8 downto 0));
    end component;

    signal LA, EA, clk, clr : STD_LOGIC := '0';
    signal dato, A : STD_LOGIC_VECTOR (8 downto 0) := (others => '0');
begin
    reg : Registro Port map (
        LA => LA,
        EA => EA,
        clk => clk,
        clr => clr,
        dato => dato,
        A => A
    );

    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';
        dato <= "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;
```

2.3. 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 ( clk, clr, LB, EB : in STD_LOGIC;
               B : out STD_LOGIC_VECTOR (3 downto 0));
    end component;

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

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

```
32
33      process
34      begin
35          clr <= '1';
36          wait for 10 ns;
37          clr <= '0';
38          wait for 10 ns;
39          LB <= '1';
40          EB <= '0';
41          wait for 10 ns;
42          LB <= '0';
43          EB <= '1';
44          wait for 50 ns;
45          LB <= '1';
46          EB <= '0';
47          wait for 10 ns;
48          LB <= '0';
49          EB <= '1';
50          wait for 50 ns;
51          LB <= '0';
52          EB <= '0';
53          wait;
54      end process;
55  end Behavioral;
```

2.4. Cartas ASM

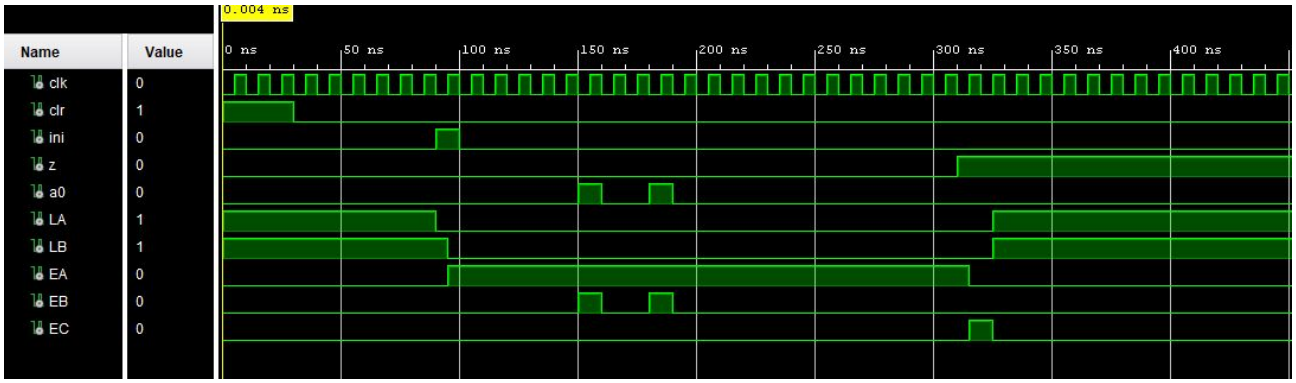
```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity tbCartasASM is
5   end tbCartasASM;
6
7   architecture Behavioral of tbCartasASM is
8       component CartasASM is
9           Port ( clk, clr, ini : in STD_LOGIC;
10                 D : in STD_LOGIC_VECTOR (8 downto 0);
11                 Dout : out STD_LOGIC_VECTOR (8 downto 0);
12                 num : out STD_LOGIC_VECTOR (3 downto 0);
13                 display : out STD_LOGIC_VECTOR (6 downto 0));
14      end component;
15
16      signal clk, clr, ini : STD_LOGIC;
```

```
17      signal D, Dout : STD_LOGIC_VECTOR (8 downto 0);
18      signal display : STD_LOGIC_VECTOR (6 downto 0);
19      signal num: STD_LOGIC_VECTOR (3 downto 0);
20  begin
21
22      asm : CartasASM Port map (
23          clk => clk,
24          clr => clr,
25          ini => ini,
26          D => D,
27          Dout => Dout,
28          num => num,
29          display => display
30      );
31
32      reloj : process begin
33          clk <= '0';
34          wait for 5 ns;
35          clk <= '1';
36          wait for 5 ns;
37      end process;
38
39      process
40      begin
41          ini <= '0';
42          clr <= '1';
43        wait for 10 ns;
44
45        clr <= '0';
46        D <= "101101011";
47        wait for 10 ns;
48
49        ini <= '1';
50        wait for 110 ns;
51
52        ini <= '0';
53        clr <= '1';
54        wait for 10 ns;
55
56        clr <= '0';
57        D <= "000011101";
58        wait for 10 ns;
59
60        ini <= '1';
```

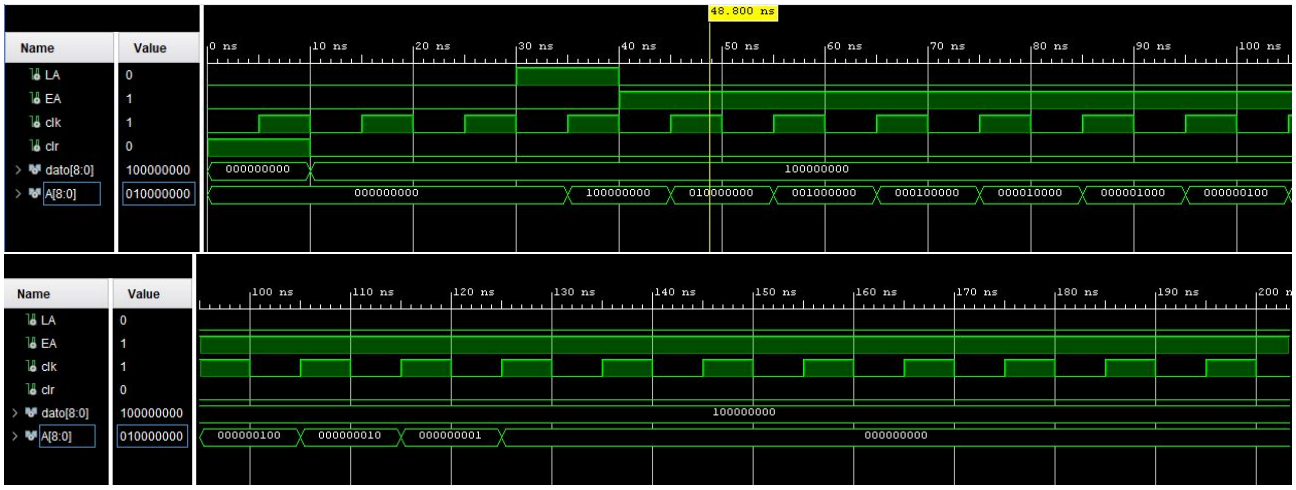
```
61        wait for 110 ns;
62
63        ini <= '0';
64        clr <= '1';
65        wait for 10 ns;
66
67        clr <= '0';
68        D <= "000010000";
69        wait for 10 ns;
70
71        ini <= '1';
72        wait for 110 ns;
73
74        ini <= '0';
75        clr <= '1';
76        wait for 10 ns;
77
78        clr <= '0';
79        D <= "100001000";
80        wait for 10 ns;
81
82        ini <= '1';
83        wait for 110 ns;
84
85        ini <= '0';
86        clr <= '1';
87        wait for 10 ns;
88
89        clr <= '0';
90        D <= "000000000";
91        wait for 10 ns;
92
93        ini <= '1';
94        wait;
95    end process;
96 end Behavioral;
```

3. Simulación

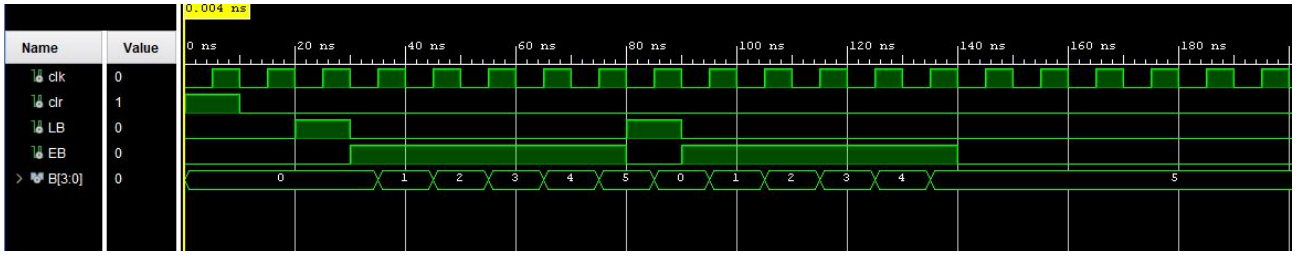
3.1. Unidad de Control



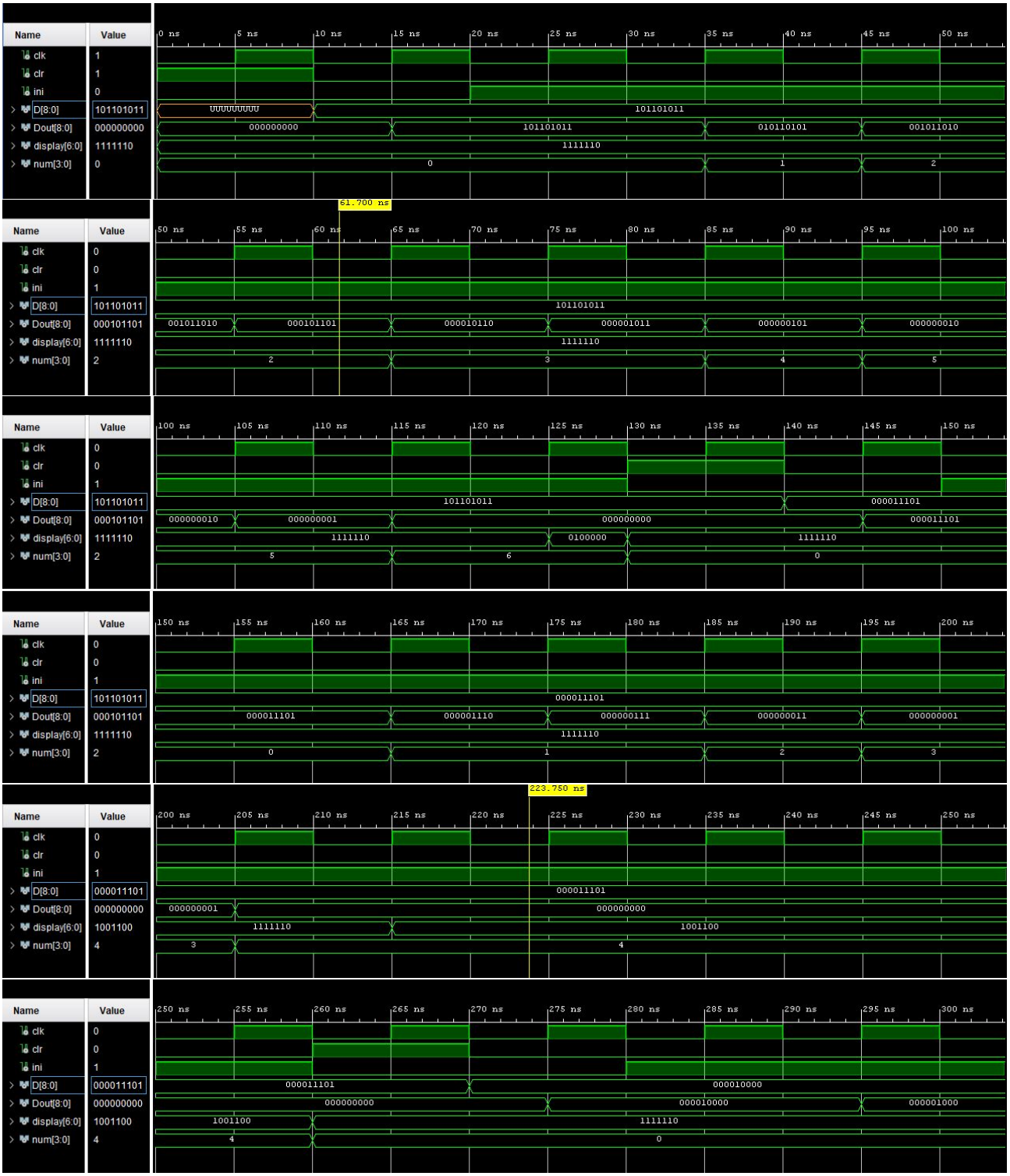
3.2. Registro

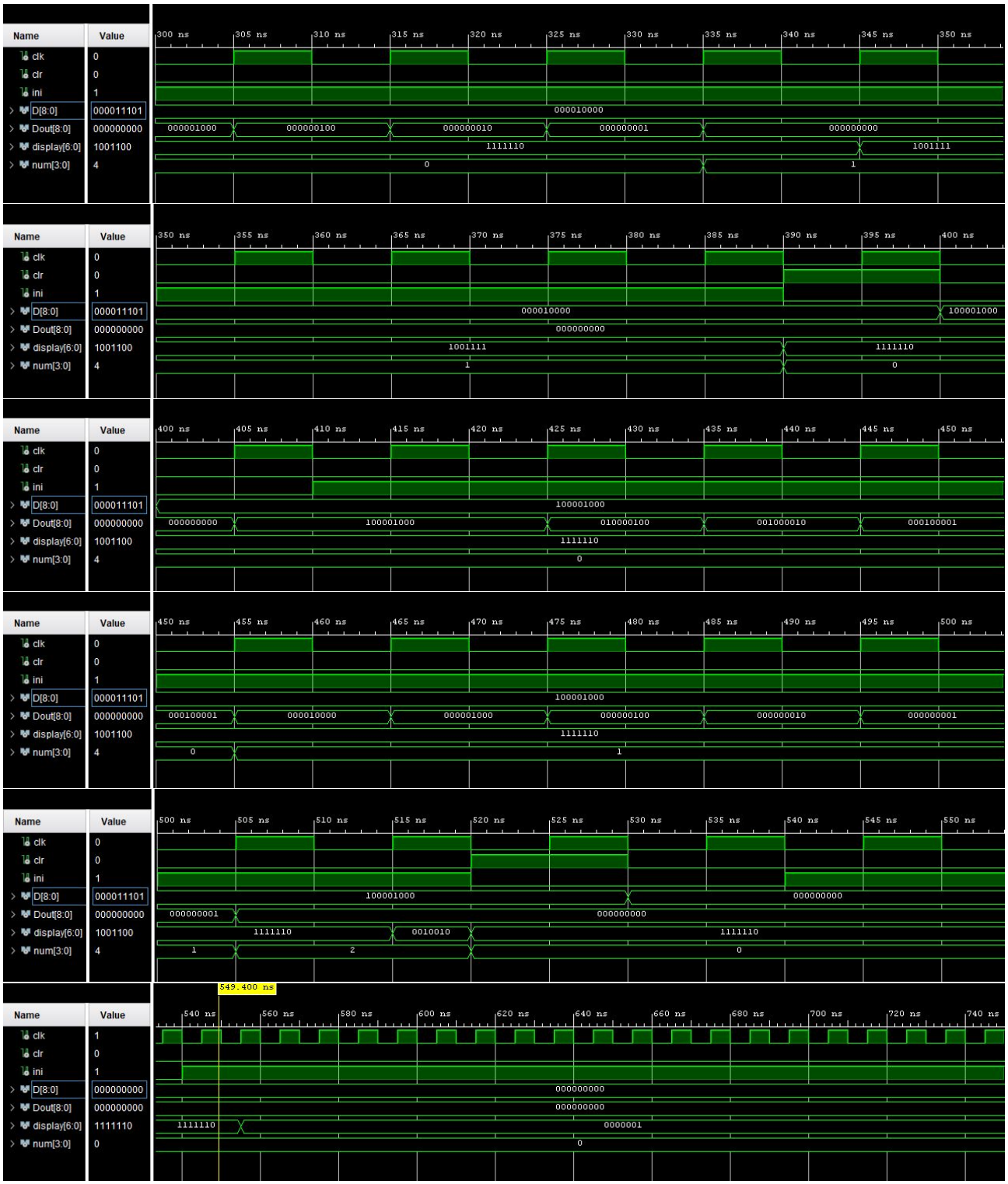


3.3. Contador



3.4. Cartas ASM

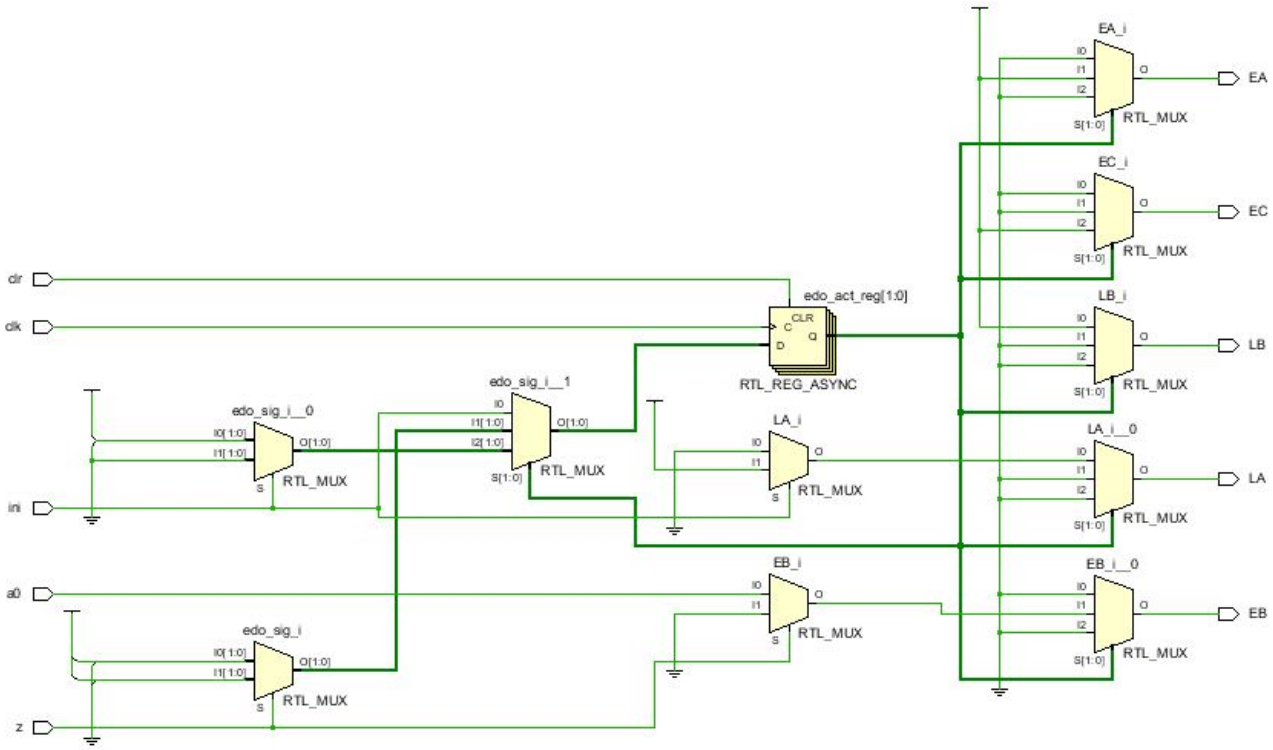



Name
Value
clk
clr
ini
D[8:0]
Dout[8:0]
display[6:0]
num[3:0]
000011101
000000000
1001100
4
300 ns
305 ns
310 ns
315 ns
320 ns
325 ns
330 ns
335 ns
340 ns
345 ns
350 ns
000010000
000001000
000000100
000000010
000000001
000000000
1111110
1001111
0
1
350 ns
355 ns
360 ns
365 ns
370 ns
375 ns
380 ns
385 ns
390 ns
395 ns
400 ns
000010000
100001000
000000000
1001111
1111110
1
0
400 ns
405 ns
410 ns
415 ns
420 ns
425 ns
430 ns
435 ns
440 ns
445 ns
450 ns
100001000
000000000
100001000
010000100
001000010
000100001
1111110
0
450 ns
455 ns
460 ns
465 ns
470 ns
475 ns
480 ns
485 ns
490 ns
495 ns
500 ns
100001000
000100001
000010000
000001000
000000100
000000010
000000001
1111110
0
1
500 ns
505 ns
510 ns
515 ns
520 ns
525 ns
530 ns
535 ns
540 ns
545 ns
550 ns
100001000
000000000
000000001
000000000
1111110
0010010
1111110
1
2
0
549.400 ns
540 ns
560 ns
580 ns
600 ns
620 ns
640 ns
660 ns
680 ns
700 ns
720 ns
740 ns
000000000
000000000
1111110
0000001
0

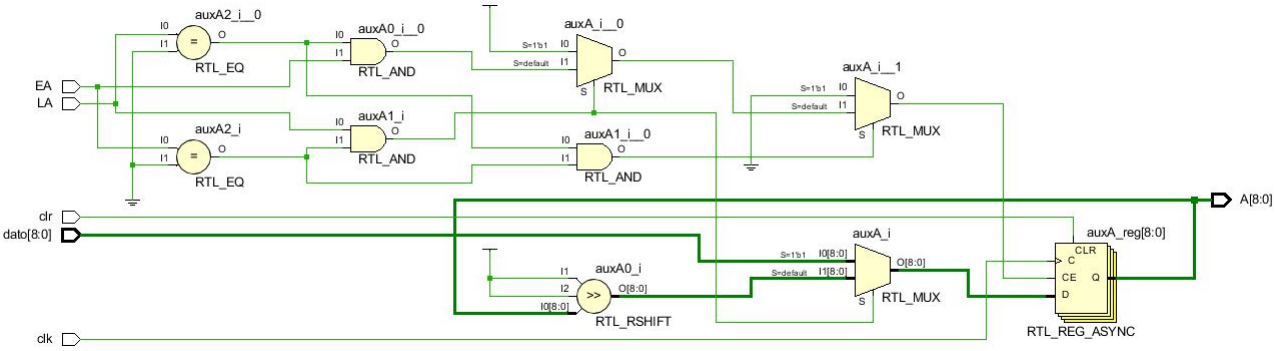
4. Diagramas RTL

4.1. Análisis RTL

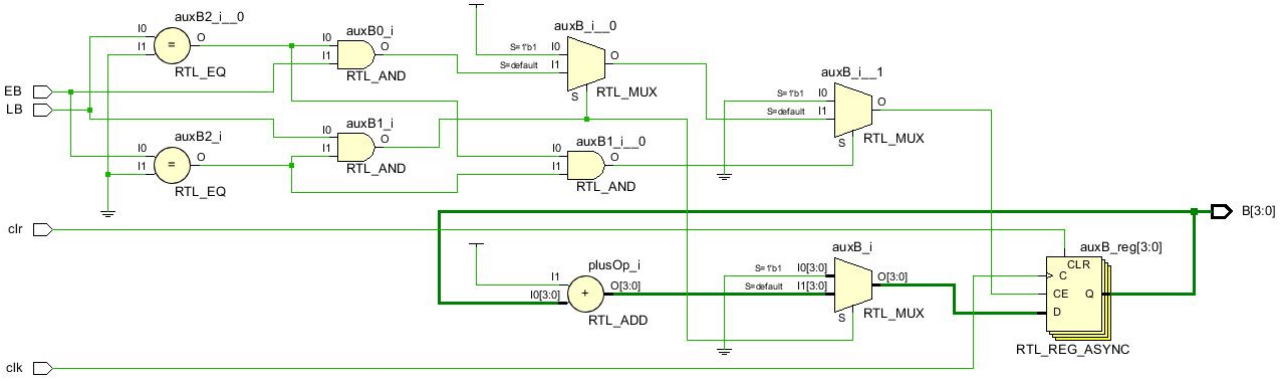
4.1.1. Unidad de Control



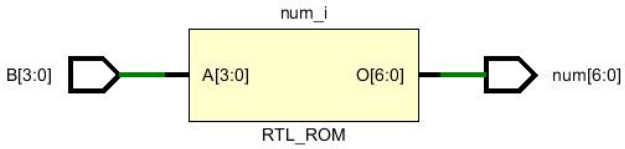
4.1.2. Registro



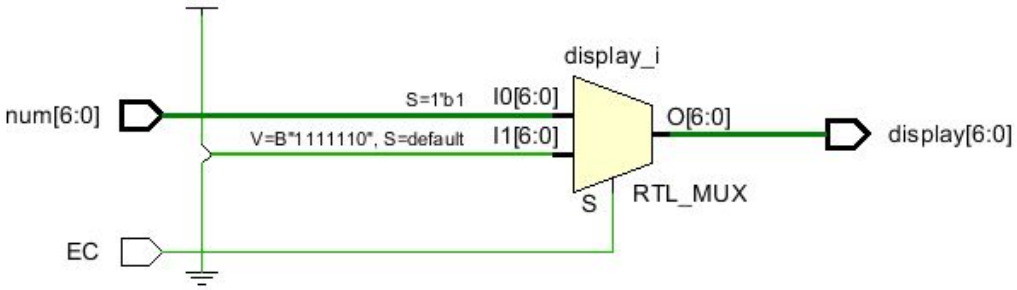
4.1.3. Contador



4.1.4. Decodificador



4.1.5. Multiplexor



4.1.6. Cartas ASM

Diagrama comprimido

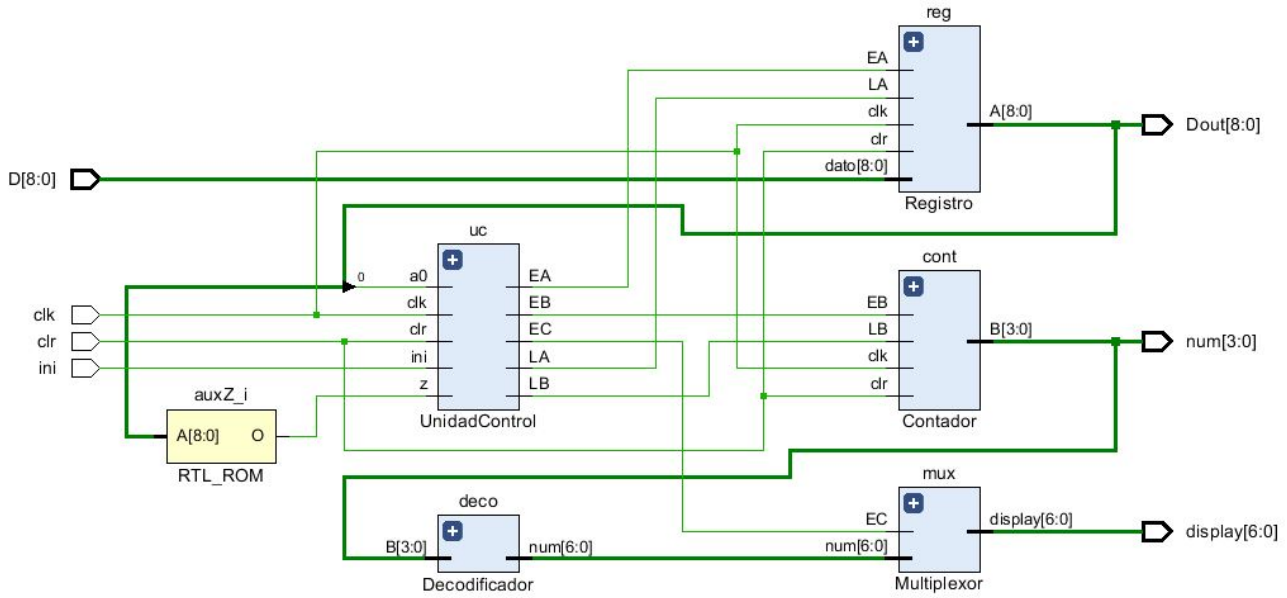
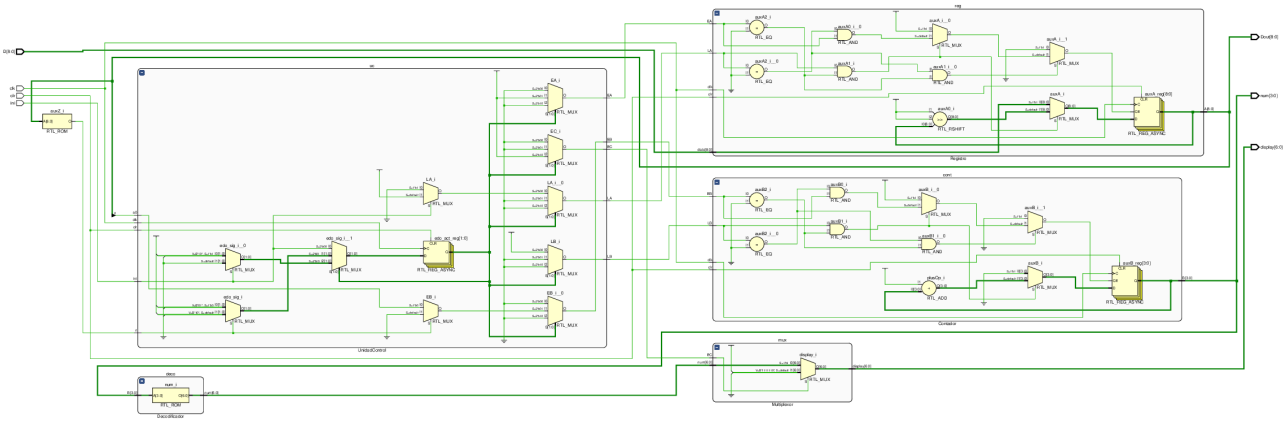
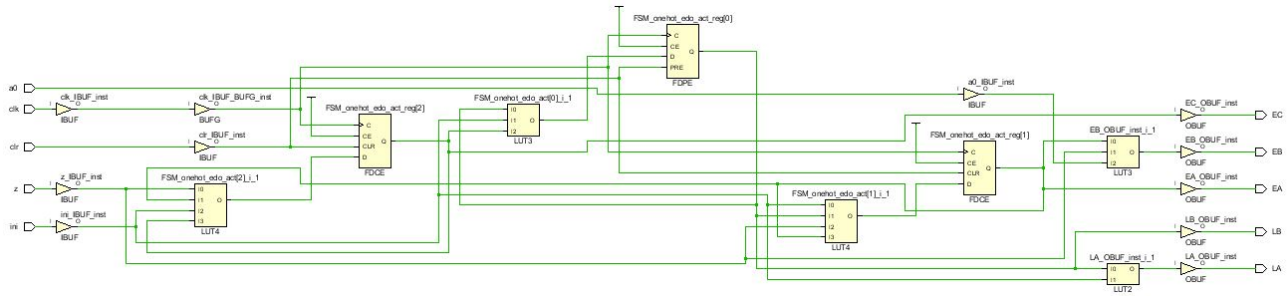


Diagrama expandido

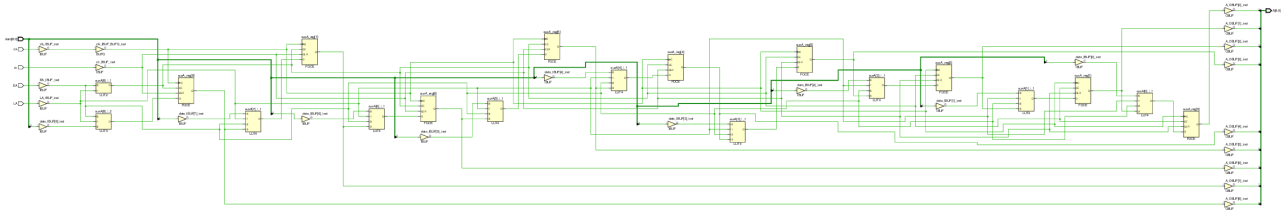


4.2. Synthesis

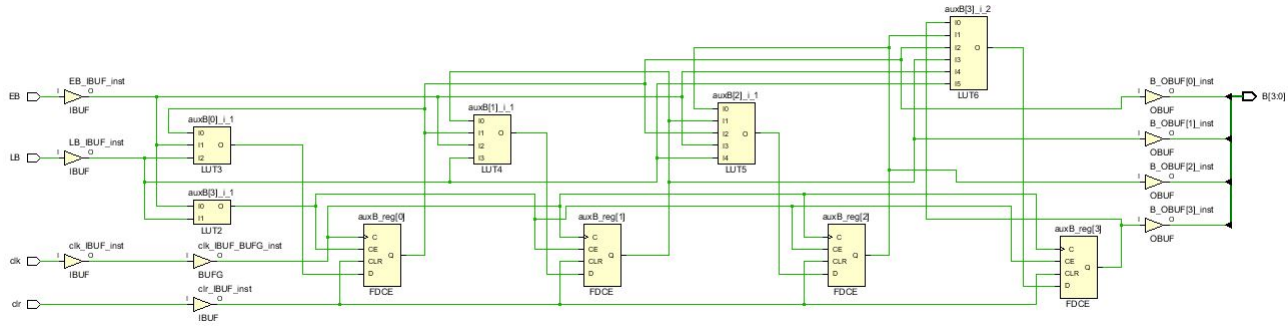
4.2.1. Unidad de Control



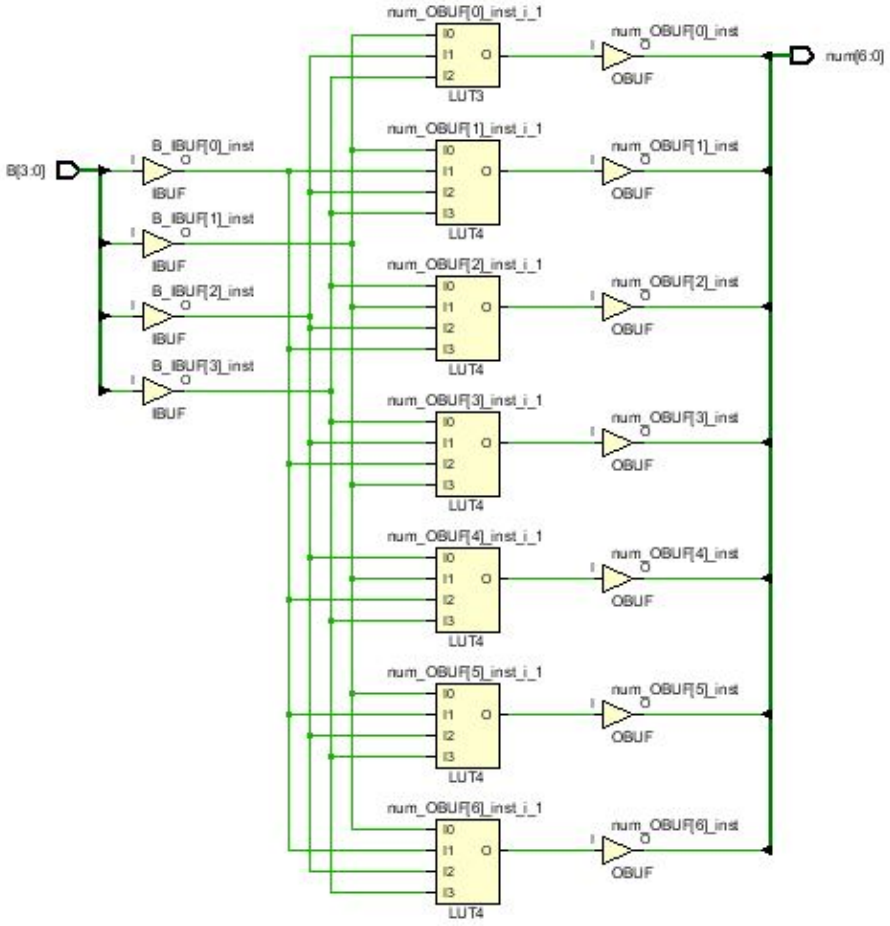
4.2.2. Registro



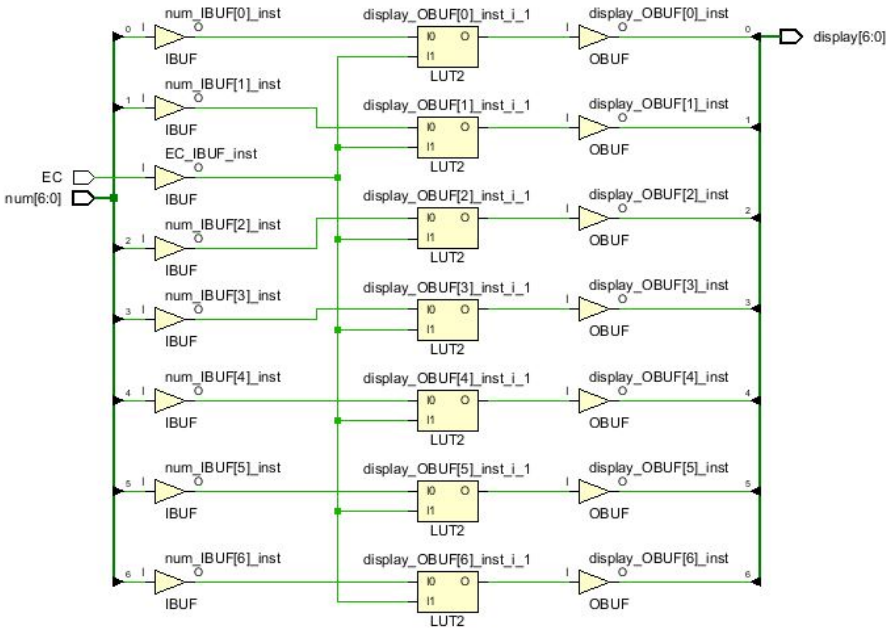
4.2.3. Contador



4.2.4. Decodificador



4.2.5. Multiplexor



4.2.6. Cartas ASM

Diagrama comprimido

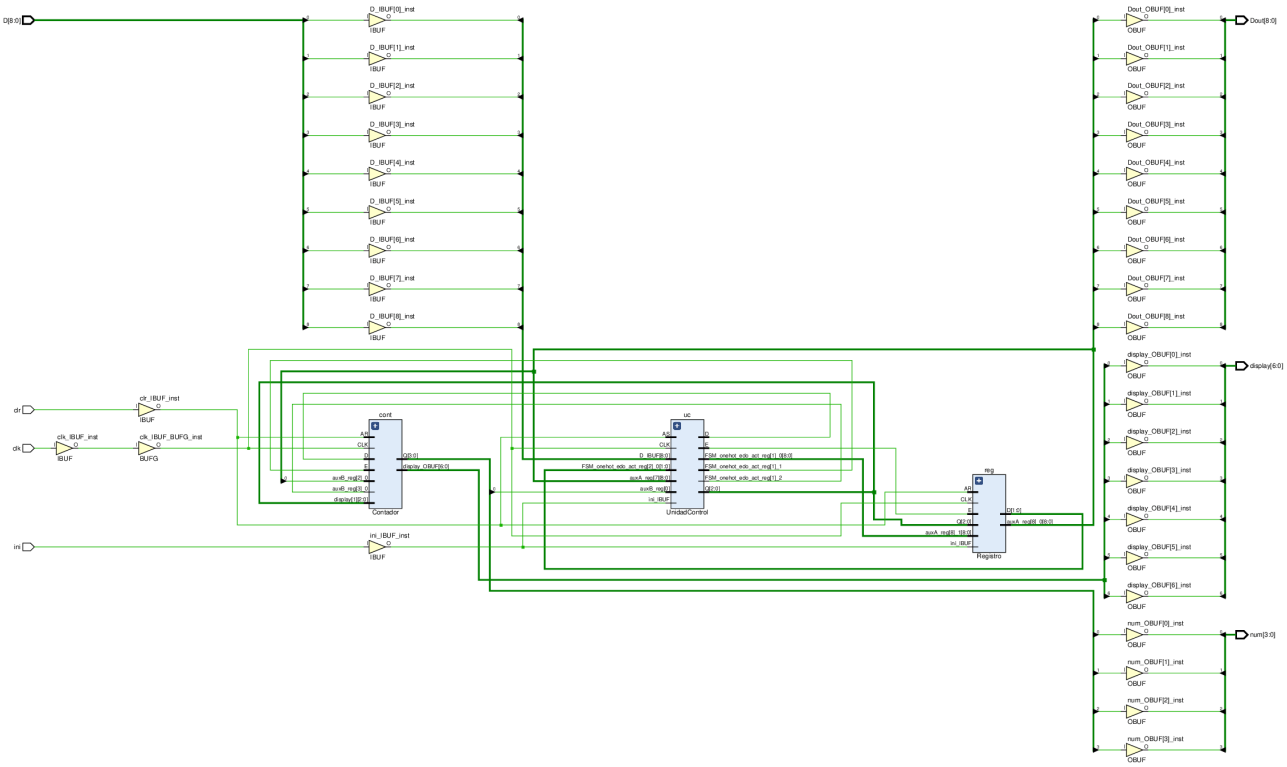


Diagrama expandido