

# Reporte de practica 14

González Pardo Adrian

Junio 2020

## 1. Código fuente de los componentes:

### 1.1. Registro

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity registro is
5     Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
6           Q : out STD_LOGIC_VECTOR (3 downto 0);
7           CLR,CLK,L : in STD_LOGIC);
8 end registro;
9
10 architecture Behavioral of registro is
11 begin
12     process(CLR,CLK,L)
13     begin
14         if CLR='1' then
15             Q<=(others=>'0');
16         elsif rising_edge(CLK) then
17             if L='1' then
18                 Q<=D;
19             end if;
20         end if;
21     end process;
22 end Behavioral;
```

### 1.2. Bloque de condición

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity condition is
5     Port ( Q : in STD_LOGIC_VECTOR (3 downto 0);
6           EQ,NE,LT,LE,GT,GE : out STD_LOGIC);
7 end condition;
8
9 architecture Behavioral of condition is
10 begin
11     -- Banderas
12     -- Q(3) = OV
13     -- Q(2) = Z
14     -- Q(1) = C
15     -- Q(0) = N
16     process(Q)
17     begin
18         EQ<=Q(2);
19         NE<=not(Q(2));
20         LT<=not(Q(1));
21         LE<=(Q(2) or (not(Q(1))));
```

```

22         GT<=((not(Q(2)))and(Q(1)));
23         GE<=Q(1);
24     end process;
25
26
27 end Behavioral;

```

### 1.3. Bloque decodificador

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4
5  entity decInstrucciones is
6      Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
7            TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI : out STD_LOGIC);
8  end decInstrucciones;
9
10 architecture Behavioral of decInstrucciones is
11
12 begin
13     process(opCode)
14     begin
15         TIPOR<='0';
16         BEQI<='0';
17         BNEQI<='0';
18         BLTI<='0';
19         BLETI<='0';
20         BGTI<='0';
21         BGETI<='0';
22         if opCode="00000" then --TIPOR
23             TIPOR<='1';
24         elsif opCode="01101" then --BEQI
25             BEQI<='1';
26         elsif opCode="01110" then --BNEQI
27             BNEQI<='1';
28         elsif opCode="01111" then --BLTI
29             BLTI<='1';
30         elsif opCode="10000" then --BLETI
31             BLETI<='1';
32         elsif opCode="10001" then --BGTI
33             BGTI<='1';
34         elsif opCode="10010" then --BGETI
35             BGETI<='1';
36         end if;
37     end process;
38 end Behavioral;

```

### 1.4. Bloque de nivel

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity level is
5      Port ( CLR,CLK : in STD_LOGIC;
6            NA : out STD_LOGIC);
7  end level;
8
9  architecture Behavioral of level is
10     signal pclk1,nclk1:std_logic;
11 begin
12     process(CLR,CLK)
13     begin
14         if CLR='1' then

```

```

15         pclk1<='1';
16     elsif rising_edge(CLK) then
17         pclk1<=not pclk1;
18     end if;
19 end process;
20 process(CLK,CLR)
21 begin
22     if CLR='1' then
23         nclk1<='0';
24     elsif falling_edge(CLK) then
25         nclk1<=not nclk1;
26     end if;
27 end process;
28 NA<=pclk1 xor nclk1;
29 end Behavioral;

```

## 1.5. MFunCode

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5 entity mfuncode is
6     Port ( funCode : in STD_LOGIC_VECTOR (3 downto 0);
7           micro : out STD_LOGIC_VECTOR (19 downto 0));
8 end mfuncode;
9
10 architecture Behavioral of mfuncode is
11 type mem is array(0 to 15) of std_logic_vector(19 downto 0);
12 constant mfuncode: mem:=( "00000100010000110011",
13                             "00000100010001110011",
14                             "00000100011100000011",
15                             "00000100011100010011",
16                             "00000100011100100011",
17                             "00000100011111010011",
18                             "00000100011111000011",
19                             "00000100011110100011",
20                             "00000100011110100011",
21                             "00000001110000000000",
22                             "00000001010000000000",
23                             others=>X"00000");
24 begin
25     micro<=mfuncode(conv_integer(funCode));
26 end Behavioral;

```

## 1.6. MOpCode

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5 entity mopcode is
6     Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
7           micro : out STD_LOGIC_VECTOR (19 downto 0));
8 end mopcode;
9
10 architecture Behavioral of mopcode is
11 type mem is array(0 to 31) of std_logic_vector(19 downto 0);
12 constant mopcode: mem:=( "00001000000001110001",-- 0 VERIFI
13                             "00000000010000000000",-- 1 LI
14                             "00000100010000000100",-- 2 LWI
15                             "00001000000000000110",-- 3 SWI
16                             "00001010000100110101",-- 4 SW
17                             "00000100010100110011",-- 5 ADDI

```

```

18         "00000100010101110011",-- 6 SUBI
19         "00000100010100000011",-- 7 ANDI
20         "00000100010100010011",-- 8 ORI
21         "00000100010100100011",-- 9 XORI
22         "00000100010111010011",-- 10 NANDI
23         "00000100010111000011",-- 11 NORI
24         "00000100010110100011",-- 12 XNORI
25         "10010000001100110011",-- 13 BEQI
26         "10010000001100110011",-- 14 BNEI
27         "10010000001100110011",-- 15 BLTI
28         "10010000001100110011",-- 16 BLETI
29         "10010000001100110011",-- 17 BGTI
30         "10010000001100110011",-- 18 BGETI
31         "00010000000000000000",-- 19 B
32         "01010000000000000000",-- 20 CALL
33         "00100000000000000000",-- 21 RET
34         "00000000000000000000",-- 22 NOP
35         "00001110010100110001", --23 LW
36         others=>x"00000");
37 begin
38     micro<=mopcode(conv_integer(opCode));
39 end Behavioral;

```

## 1.7. Unidad de Control

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity unControl is
5     Port ( TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI,NA,EQ,NE,LT,LE,GT,GE : in STD_LOGIC;
6           SM,SDOPC : out STD_LOGIC);
7 end unControl;
8
9 architecture Behavioral of unControl is
10 begin
11     SDOPC<=((not NA)and
12             ((BEQI and EQ)or(BNEQI and NE)or(BLTI and LT)or(BLETI and LE)or
13              (BGTI and GT)or(BGETI and GE)))or
14             (not(TIPOR or BEQI or BNEQI or BLTI or BLETI or BGTI or BGETI)));
15     SM<=not(TIPOR);
16 end Behavioral;

```

## 2. Código fuente del empaquetado

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 package paqueteHardware is
5
6     component condition is
7     Port ( Q : in STD_LOGIC_VECTOR (3 downto 0);
8           EQ,NE,LT,LE,GT,GE : out STD_LOGIC);
9     end component;
10
11     component decInstrucciones is
12     Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
13           TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI : out STD_LOGIC);
14     end component;
15
16     component level is
17     Port ( CLR,CLK : in STD_LOGIC;
18           NA : out STD_LOGIC);
19     end component;

```

```

20
21 component mopcode is
22 Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
23       micro : out STD_LOGIC_VECTOR (19 downto 0));
24 end component;
25
26 component mfuncode is
27 Port ( funCode : in STD_LOGIC_VECTOR (3 downto 0);
28       micro : out STD_LOGIC_VECTOR (19 downto 0));
29 end component;
30
31 component registro is
32 Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
33       Q : out STD_LOGIC_VECTOR (3 downto 0);
34       CLR,CLK,L : in STD_LOGIC);
35 end component;
36
37 component unControl is
38 Port ( TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI,NA,EQ,NE,LT,LE,GT,GE : in STD_LOGIC;
39       SM,SDOPC : out STD_LOGIC);
40 end component;
41
42 end paqueteHardware;

```

### 3. Código fuente de la unión para la arquitectura completa

```

1 library IEEE;
2 library WORK;
3 use IEEE.STD_LOGIC_1164.ALL;
4 use WORK.paqueteHardware.ALL;
5
6 entity arquitectura is
7   Port ( funCode,banderas : in STD_LOGIC_VECTOR (3 downto 0);
8         CLK,CLR : in STD_LOGIC;
9         opCode : in STD_LOGIC_VECTOR (4 downto 0);
10        microinstruccion : out STD_LOGIC_VECTOR (19 downto 0));
11 end arquitectura;
12
13 architecture Behavioral of arquitectura is
14   signal outMicro:std_logic_vector(19 downto 0);
15   signal muxMopCode:std_logic_vector(4 downto 0);
16   signal outMFunCode,outMOpCode:std_logic_vector(19 downto 0);
17   signal NA,EQ,NE,LT,LE,GT,GE,LF,SM,SDOPC,TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI:
18   std_logic;
19   signal Q:std_logic_vector(3 downto 0);
20 begin
21   process(SDOPC,SM,outMicro,opCode,outMOpCode,outMFunCode)
22   begin
23     if SDOPC='1' then
24       muxMopCode<=opCode;
25     else
26       muxMopCode<=(others=>'0');
27     end if;
28
29     if SM='1' then
30       outMicro<=outMOpCode;
31     else
32       outMicro<=outMFunCode;
33     end if;
34     LF<=outMicro(0);
35     microinstruccion<=outMicro;
36   end process;
37
38   mfuncode_co:mfuncode port map(

```

```

39         funCode=>funCode ,
40         micro=>outMFunCode
41     );
42
43     decodificador:decInstrucciones port map(
44         opCode=>opCode ,
45         TIPOR=>TIPOR ,
46         BEQI=>BEQI ,
47         BNEQI=>BNEQI ,
48         BLTI=>BLTI ,
49         BLETI=>BLETI ,
50         BGTI=>BGTI ,
51         BGETI=>BGETI
52     );
53
54     level_com:level port map(
55         CLR=>CLR ,
56         CLK=>CLK ,
57         NA=>NA
58     );
59
60     reg_com:registro port map(
61         D=>banderas ,
62         Q=>Q ,
63         CLR=>CLR ,
64         CLK=>CLK ,
65         L=>LF
66     );
67
68     condition_com:condition port map(
69         Q=>Q ,
70         EQ=>EQ ,
71         NE=>NE ,
72         LT=>LT ,
73         LE=>LE ,
74         GT=>GT ,
75         GE=>GE
76     );
77
78     mop_com:mopcode port map(
79         opCode=>muxMopCode ,
80         micro=>outMOpCode
81     );
82
83     unC_com:unControl port map(
84         TIPOR=>TIPOR ,
85         BEQI=>BEQI ,
86         BNEQI=>BNEQI ,
87         BLTI=>BLTI ,
88         BLETI=>BLETI ,
89         BGTI=>BGTI ,
90         BGETI=>BGETI ,
91         NA=>NA ,
92         EQ=>EQ ,
93         NE=>NE ,
94         LT=>LT ,
95         LE=>LE ,
96         GT=>GT ,
97         GE=>GE ,
98         SM=>SM ,
99         SDOPC=>SDOPC
100    );
101
102 end Behavioral;

```

## 4. Test-Bench:

### 4.1. Registro

```
1 library IEEE;
2 library WORK;
3 use IEEE.STD_LOGIC_1164.ALL;
4 use WORK.paqueteHardware.registro;
5
6 entity registro_TB is
7 end registro_TB;
8
9 architecture Behavioral of registro_TB is
10     signal CLK,CLR,L:std_logic:='0';
11     signal D,Q:std_logic_vector(3 downto 0):=(others=>'0');
12     constant CLK_P:time:=10ns;
13 begin
14     reg_comp:registro port map(
15         D=>D,
16         Q=>Q,
17         CLR=>CLR,
18         CLK=>CLK,
19         L=>L
20     );
21     clk_Pr:process
22     begin
23         wait for CLK_P/2;
24         CLK<='1';
25         wait for CLK_P/2;
26         CLK<='0';
27     end process;
28
29     regTB:process
30     begin
31         CLR<='1';
32         wait for CLK_P;
33         CLR<='0';
34         D<="0100";
35         wait for CLK_P;
36         L<='1';
37         wait for CLK_P;
38         D<="1100";
39         wait for CLK_P;
40         L<='0';
41         D<="0101";
42         wait for CLK_P;
43         wait;
44     end process;
45 end Behavioral;
```

### 4.2. Bloque de condición

```
1 library IEEE;
2 library WORK;
3 use IEEE.STD_LOGIC_1164.ALL;
4 use WORK.paqueteHardware.condition;
5
6 entity condition_TB is
7 end condition_TB;
8
9 architecture Behavioral of condition_TB is
10     signal Q:STD_LOGIC_VECTOR (3 downto 0);
11     signal EQ,NE,LT,LE,GT,GE:STD_LOGIC;
12 begin
13     condition_com:condition port map(
```

```

14      Q=>Q,
15      EQ=>EQ,
16      NE=>NE,
17      LT=>LT,
18      LE=>LE,
19      GT=>GT,
20      GE=>GE
21 );
22 condition_TB:process
23 begin
24     Q<="0000";
25     wait for 10 ns;
26     Q<="0001";
27     wait for 10 ns;
28     Q<="0010";
29     wait for 10 ns;
30     Q<="0011";
31     wait for 10 ns;
32     Q<="0100";
33     wait for 10 ns;
34     Q<="0101";
35     wait for 10 ns;
36     Q<="0110";
37     wait;
38 end process;
39
40
41 end Behavioral;

```

### 4.3. Bloque decodificador

```

1  library IEEE;
2  library WORK;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use WORK.paqueteHardware.decInstrucciones;
5
6  entity decodificador_TB is
7  end decodificador_TB;
8
9  architecture Behavioral of decodificador_TB is
10     signal opCode:STD_LOGIC_VECTOR (4 downto 0):=(others =>'0');
11     signal TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI:STD_LOGIC:='0';
12 begin
13     decodificador_com:decInstrucciones port map(
14         opCode=>opCode,
15         TIPOR=>TIPOR,
16         BEQI=>BEQI,
17         BNEQI=>BNEQI,
18         BLTI=>BLTI,
19         BLETI=>BLETI,
20         BGTI=>BGTI,
21         BGETI=>BGETI
22     );
23
24     decInstrucciones_TB:process
25     begin
26         opCode <= "00000";
27         wait for 10 ns;
28         opCode <= "01101";
29         wait for 10 ns;
30         opCode <= "01110";
31         wait for 10 ns;
32         opCode <= "01111";
33         wait for 10 ns;
34         opCode <= "10000";
35         wait for 10 ns;

```



```

36         opCode <= "10001";
37         wait for 10 ns;
38         opCode <= "10010";
39         wait;
40     end process;
41
42 end Behavioral;

```

## 4.4. Bloque de nivel

```

1  library IEEE;
2  library WORK;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use WORK.paqueteHardware.level;
5
6  entity level_TB is
7  end level_TB;
8
9  architecture Behavioral of level_TB is
10     signal CLR,CLK,NA:std_logic:='0';
11     constant CLK_P:time:=10ns;
12 begin
13     level_com:level port map(
14         CLR=>CLR,
15         CLK=>CLK,
16         NA=>NA
17     );
18
19     clk_Pr:process
20     begin
21         wait for CLK_P/2;
22         CLK<='1';
23         wait for CLK_P/2;
24         CLK<='0';
25     end process;
26
27     levelTB:process
28     begin
29         CLR <= '1';
30         wait for CLK_P;
31         CLR <= '0';
32         wait for CLK_P;
33         wait for CLK_P;
34         wait for CLK_P;
35         wait;
36     end process;
37
38
39 end Behavioral;

```

## 4.5. MFunCode

```

1  library IEEE;
2  library WORK;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use WORK.paqueteHardware.mfuncode;
5
6  entity mfuncode_TB is
7  end mfuncode_TB;
8
9  architecture Behavioral of mfuncode_TB is
10     signal funCode:STD_LOGIC_VECTOR (3 downto 0):=(others=>'0');
11     signal micro:STD_LOGIC_VECTOR (19 downto 0):=(others=>'0');
12 begin

```

```

13     mfuncode_com:mfuncode port map(
14         funCode=>funCode,
15         micro=>micro
16     );
17
18     mfuncode_TB:process
19     begin
20         funCode <= "0000";
21         wait for 10 ns;
22         funCode <= "0001";
23         wait for 10 ns;
24         funCode <= "0010";
25         wait for 10 ns;
26         funCode <= "0011";
27         wait for 10 ns;
28         funCode <= "0100";
29         wait for 10 ns;
30         funCode <= "0101";
31         wait for 10 ns;
32         funCode <= "0110";
33         wait for 10 ns;
34         funCode <= "0111";
35         wait for 10 ns;
36         funCode <= "1000";
37         wait for 10 ns;
38         funCode <= "1001";
39         wait for 10 ns;
40         funCode <= "1010";
41         wait for 10 ns;
42         funCode <= "1011";
43         wait for 10 ns;
44         funCode <= "1100";
45         wait for 10 ns;
46         funCode <= "1101";
47         wait for 10 ns;
48         funCode <= "1110";
49         wait for 10 ns;
50         funCode <= "1111";
51         wait;
52     end process;
53
54
55 end Behavioral;

```

## 4.6. MOpCode

```

1  library IEEE;
2  library WORK;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use WORK.paqueteHardware.mopcode;
5
6  entity mopcode_TB is
7  end mopcode_TB;
8
9  architecture Behavioral of mopcode_TB is
10     signal opCode:STD_LOGIC_VECTOR (4 downto 0):=(others=>'0');
11     signal micro:STD_LOGIC_VECTOR (19 downto 0):=(others=>'0');
12 begin
13     mopcode_com:mopcode port map(
14         opCode=>opCode,
15         micro=>micro
16     );
17     mopcode_TB:process
18     begin
19         opCode <= "00000";
20         wait for 10 ns;

```

```

21     opCode <= "00001";
22     wait for 10 ns;
23     opCode <= "00010";
24     wait for 10 ns;
25     opCode <= "00011";
26     wait for 10 ns;
27     opCode <= "00100";
28     wait for 10 ns;
29     opCode <= "00101";
30     wait for 10 ns;
31     opCode <= "00110";
32     wait for 10 ns;
33     opCode <= "00111";
34     wait for 10 ns;
35     opCode <= "01000";
36     wait for 10 ns;
37     opCode <= "01001";
38     wait for 10 ns;
39     opCode <= "01010";
40     wait for 10 ns;
41     opCode <= "01011";
42     wait for 10 ns;
43     opCode <= "01100";
44     wait for 10 ns;
45     opCode <= "01101";
46     wait for 10 ns;
47     opCode <= "01110";
48     wait for 10 ns;
49     opCode <= "01111";
50     wait;
51 end process;
52 end Behavioral;

```

## 4.7. Unidad de Control

```

1  library IEEE;
2  library WORK;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use WORK.paqueteHardware.unControl;
5
6  entity unControl_TB is
7  end unControl_TB;
8
9  architecture Behavioral of unControl_TB is
10     signal TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI,NA,EQ,NE,LT,LE,GT,GE,SM,SDOPC:STD_LOGIC
       := '0';
11     constant CLK_P:time:=10ns;
12 begin
13     unControl_com:unControl port map(
14         TIPOR=>TIPOR,
15         BEQI=>BEQI,
16         BNEQI=>BNEQI,
17         BLTI=>BLTI,
18         BLETI=>BLETI,
19         BGTI=>BGTI,
20         BGETI=>BGETI,
21         NA=>NA,EQ=>EQ,
22         NE=>NE,
23         LT=>LT,
24         LE=>LE,
25         GT=>GT,
26         GE=>GE,
27         SM=>SM,
28         SDOPC=>SDOPC
29     );
30     clk_Pr:process

```

```

31 begin
32     wait for CLK_P/2;
33     NA<='1';
34     wait for CLK_P/2;
35     NA<='0';
36 end process;
37
38 unControl_TB:process
39 begin
40     TIPOR<='0';
41     BEQI<='0';
42     BNEQI<='0';
43     BLTI<='0';
44     BLETI<='0';
45     BGTI<='0';
46     BGETI<='0';
47     EQ<='0';
48     NE<='0';
49     LT<='0';
50     LE<='0';
51     GT<='0';
52     GE<='0';
53     wait for CLK_P;
54     TIPOR<='1';
55     wait for CLK_P;
56     TIPOR<='0';
57     wait for CLK_P;
58     BEQI<='1';
59     EQ<='1';
60     wait for CLK_P;
61     BEQI<='0';
62     EQ<='0';
63     BNEQI<='1';
64     NE<='0';
65     wait for CLK_P;
66     wait;
67 end process;
68 end Behavioral;

```

## 4.8. Arquitectura completa

```

1  library IEEE;
2  LIBRARY STD;
3  use IEEE.STD_LOGIC_1164.ALL;
4  USE STD.TEXTIO.ALL;
5  USE ieee.std_logic_TEXTIO.ALL;  --PERMITE USAR STD_LOGIC
6  USE ieee.std_logic_UNSIGNED.ALL;
7  USE ieee.std_logic_ARITH.ALL;
8
9
10 entity arquitectura_TB is
11 end arquitectura_TB;
12
13 architecture Behavioral of arquitectura_TB is
14     component arquitectura is
15     Port ( funCode,banderas : in STD_LOGIC_VECTOR (3 downto 0);
16           CLK,CLR : in STD_LOGIC;
17           opCode : in STD_LOGIC_VECTOR (4 downto 0);
18           microinstruccion : out STD_LOGIC_VECTOR (19 downto 0));
19     end component;
20     signal funCode,banderas : STD_LOGIC_VECTOR (3 downto 0):=(others=>'0');
21     signal CLK,CLR : STD_LOGIC:='0';
22     signal opCode : STD_LOGIC_VECTOR (4 downto 0):=(others=>'0');
23     signal microinstruccion : STD_LOGIC_VECTOR (19 downto 0):=(others=>'0');
24     constant CLK_P : time:=10 ns;
25 begin

```

```

26  arquitectura_TB:arquitectura port map(
27      funCode=>funCode,
28      banderas=>banderas,
29      CLK=>CLK,
30      CLR=>CLR,
31      opCode=>opCode,
32      microinstruccion=>microinstruccion
33  );
34
35  clk_Pr:process
36  begin
37      CLK<='1';
38      wait for CLK_P/2;
39      CLK<='0';
40      wait for CLK_P/2;
41  end process;
42
43  simulacion:process
44      --salida
45      file ARCH_SAL : TEXT;
46      variable LINEA_SAL: line;
47      variable VAR_Microins: STD_LOGIC_VECTOR(19 downto 0);
48
49      --entrada
50      file ARCH_ENT :TEXT;
51      variable LINEA_ENT :line;
52      VARIABLE VAR_FunCode : STD_LOGIC_VECTOR (3 downto 0);
53      VARIABLE VAR_opCode : STD_LOGIC_VECTOR (4 downto 0);
54      VARIABLE VAR_CLK,VAR_CLR,VAR_LF : STD_LOGIC;
55      VARIABLE VAR_Flags : STD_LOGIC_VECTOR (3 downto 0);
56
57      VARIABLE CADENA : STRING(1 TO 9);
58      VARIABLE NIVEL : STRING(1 TO 4);
59  begin
60      file_open(ARCH_ENT, "/run/media/d3vcr4ck/externData/materias-Sem20_2/
arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/unidadDeControl/
in.txt", READ_MODE);
61      file_open(ARCH_SAL, "/run/media/d3vcr4ck/externData/materias-Sem20_2/
arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/unidadDeControl/
out.txt", WRITE_MODE);
62
63      CADENA := "_OP_CODE_";
64      write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
65      CADENA := "_FUN_CODE";
66      write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
67      CADENA := "_BANDERAS";
68      write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
69      CADENA := " __CLR__";
70      write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
71      --CADENA := " ___LF___";
72      --write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
73      CADENA := "_MICRONST";
74      write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
75      CADENA := " __NIVEL_";
76      write(LINEA_SAL, CADENA, right, CADENA'LENGTH+11);
77
78      writeline(ARCH_SAL,LINEA_SAL);
79
80      wait for 100ns;
81      for i in 0 to 51 loop
82          readline(ARCH_ENT,LINEA_ENT);
83
84          read(LINEA_ENT,VAR_opCode);
85          opCode <= VAR_opCode;
86
87          read(LINEA_ENT,VAR_FunCode);
88          funCode <= VAR_FunCode;

```

```

89      read(LINEA_ENT,VAR_Flags);
90      banderas <= VAR_Flags;
91
92
93      read(LINEA_ENT,VAR_CLR);
94      CLR <= VAR_CLR;
95
96      read(LINEA_ENT,VAR_LF);
97
98      --wait until rising_edge(CLK);
99      wait for 5ns;
100     VAR_Microins:=microinstruccion;
101
102     WRITE(LINEA_SAL,VAR_opCode, right, 5);
103     WRITE(LINEA_SAL,VAR_FunCode, right, 10);
104     WRITE(LINEA_SAL,VAR_Flags, right, 10);
105     WRITE(LINEA_SAL,VAR_CLR, right, 10);
106     WRITE(LINEA_SAL,VAR_Microins, right, 26);
107     if CLK = '1' then
108         NIVEL := "ALTO";
109         write(LINEA_SAL, NIVEL, right, NIVEL'LENGTH+3);
110     else
111         NIVEL := "BAJO";
112         write(LINEA_SAL, NIVEL, right, NIVEL'LENGTH+3);
113     end if;
114     writeline(ARCH_SAL,LINEA_SAL);
115
116     wait for 5ns;
117     VAR_Microins:=microinstruccion;
118
119     WRITE(LINEA_SAL,VAR_opCode, right, 5);
120     WRITE(LINEA_SAL,VAR_FunCode, right, 10);
121     WRITE(LINEA_SAL,VAR_Flags, right, 10);
122     WRITE(LINEA_SAL,VAR_CLR, right,10);
123     WRITE(LINEA_SAL,VAR_Microins, right, 26);
124     if CLK = '1' then
125         NIVEL := "ALTO";
126         write(LINEA_SAL, NIVEL, right, NIVEL'LENGTH+3);
127     else
128         NIVEL := "BAJO";
129         write(LINEA_SAL, NIVEL, right, NIVEL'LENGTH+3);
130     end if;
131     writeline(ARCH_SAL,LINEA_SAL);
132 end loop;
133 file_close(ARCH_ENT);
134 file_close(ARCH_SAL);
135 wait;
136 end process;
137 end Behavioral;

```

## 5. Archivos de texto input/output

### 5.1. Entrada

```

1 00000 0000 0000 1 0
2 00000 0000 0000 1 0
3 00000 0000 0001 0 1
4 00000 0000 0010 0 1
5 00000 0001 0001 0 1
6 00000 0010 0100 0 1
7 00000 0011 1100 0 1
8 00000 0100 0011 0 1
9 00000 0101 1000 0 1
10 00000 0110 0001 0 1
11 00000 0111 0100 0 1

```

```

12 00000 1000 0010 0 1
13 00000 1001 0000 0 0
14 00000 1010 0000 0 0
15 00000 1011 0000 0 0
16 00000 1100 0000 0 0
17 00001 0111 0000 0 0
18 00010 0100 0000 0 0
19 00011 1000 0000 0 0
20 00100 0110 0000 0 0
21 00101 0000 0010 0 1
22 00110 0110 0001 0 1
23 00111 0100 0011 0 1
24 01000 1010 0100 0 1
25 01001 0100 1000 0 1
26 01010 0001 1100 0 1
27 01011 0011 0101 0 1
28 01100 1111 1010 0 1
29 10111 0000 0000 0 1
30 01101 1111 0000 0 1
31 01101 1011 0010 0 1
32 01101 1101 0010 0 1
33 01110 1110 0010 0 1
34 01110 1100 0000 0 1
35 01110 0011 0000 0 1
36 01111 0001 1100 0 1
37 01111 0000 1000 0 1
38 01111 0010 0100 0 1
39 10000 0100 0000 0 1
40 10000 0110 1110 0 1
41 10000 0101 1000 0 1
42 10001 0111 1010 0 1
43 10001 1010 1100 0 1
44 10001 1000 0000 0 1
45 10010 1111 1000 0 1
46 10010 1001 1010 0 1
47 10010 1101 1100 0 1
48 10011 1001 1100 0 0
49 10100 1111 0000 0 0
50 10101 0000 0000 0 0
51 10110 0000 0000 0 0
52 11000 0000 0000 0 0

```

## 5.2. Salida

	_OP_CODE_	_FUN_CODE	_BANDERAS	--CLR---	_MICRONST	--NIVEL_
1	00000	0000	0000	1	00000100010000110011	ALTO
2	00000	0000	0000	1	00000100010000110011	BAJO
3	00000	0000	0000	1	00000100010000110011	ALTO
4	00000	0000	0000	1	00000100010000110011	BAJO
5	00000	0000	0000	1	00000100010000110011	BAJO
6	00000	0000	0001	0	00000100010000110011	ALTO
7	00000	0000	0001	0	00000100010000110011	BAJO
8	00000	0000	0010	0	00000100010000110011	ALTO
9	00000	0000	0010	0	00000100010000110011	BAJO
10	00000	0001	0001	0	00000100010001110011	ALTO
11	00000	0001	0001	0	00000100010001110011	BAJO
12	00000	0010	0100	0	00000100011100000011	ALTO
13	00000	0010	0100	0	00000100011100000011	BAJO
14	00000	0011	1100	0	00000100011100010011	ALTO
15	00000	0011	1100	0	00000100011100010011	BAJO
16	00000	0100	0011	0	00000100011100100011	ALTO
17	00000	0100	0011	0	00000100011100100011	BAJO
18	00000	0101	1000	0	00000100011111010011	ALTO
19	00000	0101	1000	0	00000100011111010011	BAJO
20	00000	0110	0001	0	00000100011111000011	ALTO
21	00000	0110	0001	0	00000100011111000011	BAJO
22	00000	0111	0100	0	00000100011111010011	ALTO

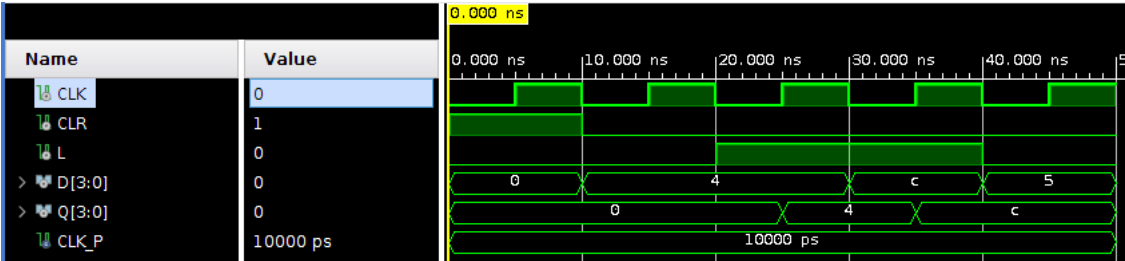
23	00000	0111	0100	0	00000100011110100011	BAJO
24	00000	1000	0010	0	00000100011111010011	ALTO
25	00000	1000	0010	0	00000100011111010011	BAJO
26	00000	1001	0000	0	00000001110000000000	ALTO
27	00000	1001	0000	0	00000001110000000000	BAJO
28	00000	1010	0000	0	00000001010000000000	ALTO
29	00000	1010	0000	0	00000001010000000000	BAJO
30	00000	1011	0000	0	00000000000000000000	ALTO
31	00000	1011	0000	0	00000000000000000000	BAJO
32	00000	1100	0000	0	00000000000000000000	ALTO
33	00000	1100	0000	0	00000000000000000000	BAJO
34	00001	0111	0000	0	00000000010000000000	ALTO
35	00001	0111	0000	0	00000000010000000000	BAJO
36	00010	0100	0000	0	000001000100000001000	ALTO
37	00010	0100	0000	0	000001000100000001000	BAJO
38	00011	1000	0000	0	000010000000000001100	ALTO
39	00011	1000	0000	0	000010000000000001100	BAJO
40	00100	0110	0000	0	00001010000100110101	ALTO
41	00100	0110	0000	0	00001010000100110101	BAJO
42	00101	0000	0010	0	00000100010100110011	ALTO
43	00101	0000	0010	0	00000100010100110011	BAJO
44	00110	0110	0001	0	00000100010101110011	ALTO
45	00110	0110	0001	0	00000100010101110011	BAJO
46	00111	0100	0011	0	00000100010100000011	ALTO
47	00111	0100	0011	0	00000100010100000011	BAJO
48	01000	1010	0100	0	00000100010100010011	ALTO
49	01000	1010	0100	0	00000100010100010011	BAJO
50	01001	0100	1000	0	00000100010100100011	ALTO
51	01001	0100	1000	0	00000100010100100011	BAJO
52	01010	0001	1100	0	00000100010111010011	ALTO
53	01010	0001	1100	0	00000100010111010011	BAJO
54	01011	0011	0101	0	00000100010111000011	ALTO
55	01011	0011	0101	0	00000100010111000011	BAJO
56	01100	1111	1010	0	00000100010110100011	ALTO
57	01100	1111	1010	0	00000100010110100011	BAJO
58	10111	0000	0000	0	00001110010100110001	ALTO
59	10111	0000	0000	0	00001110010100110001	BAJO
60	01101	1111	0000	0	00001000000001110001	ALTO
61	01101	1111	0000	0	00001000000001110001	BAJO
62	01101	1011	0010	0	00001000000001110001	ALTO
63	01101	1011	0010	0	00001000000001110001	BAJO
64	01101	1101	0010	0	00001000000001110001	ALTO
65	01101	1101	0010	0	00001000000001110001	BAJO
66	01110	1110	0010	0	10010000001100110011	ALTO
67	01110	1110	0010	0	00001000000001110001	BAJO
68	01110	1100	0000	0	10010000001100110011	ALTO
69	01110	1100	0000	0	00001000000001110001	BAJO
70	01110	0011	0000	0	10010000001100110011	ALTO
71	01110	0011	0000	0	00001000000001110001	BAJO
72	01111	0001	1100	0	10010000001100110011	ALTO
73	01111	0001	1100	0	00001000000001110001	BAJO
74	01111	0000	1000	0	10010000001100110011	ALTO
75	01111	0000	1000	0	00001000000001110001	BAJO
76	01111	0010	0100	0	10010000001100110011	ALTO
77	01111	0010	0100	0	00001000000001110001	BAJO
78	10000	0100	0000	0	10010000001100110011	ALTO
79	10000	0100	0000	0	00001000000001110001	BAJO
80	10000	0110	1110	0	10010000001100110011	ALTO
81	10000	0110	1110	0	00001000000001110001	BAJO
82	10000	0101	1000	0	10010000001100110011	ALTO
83	10000	0101	1000	0	00001000000001110001	BAJO
84	10001	0111	1010	0	10010000001100110011	ALTO
85	10001	0111	1010	0	00001000000001110001	BAJO
86	10001	1010	1100	0	00001000000001110001	ALTO
87	10001	1010	1100	0	00001000000001110001	BAJO
88	10001	1000	0000	0	00001000000001110001	ALTO
89	10001	1000	0000	0	00001000000001110001	BAJO



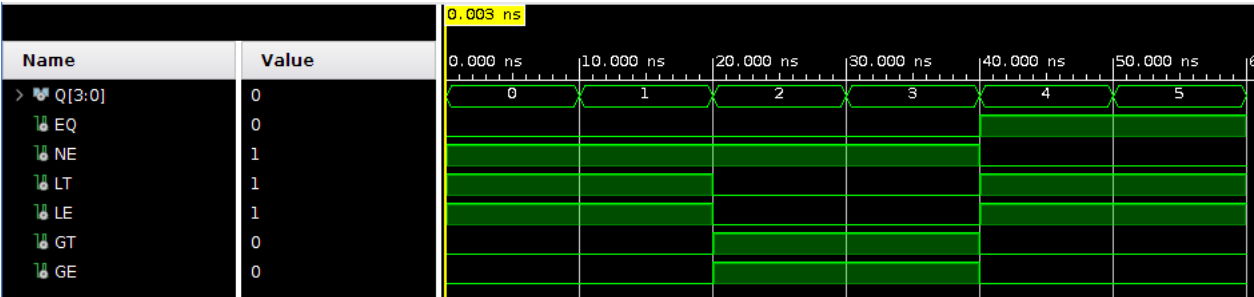
90	10010	1111	1000	0	000010000000001110001	ALTO
91	10010	1111	1000	0	000010000000001110001	BAJO
92	10010	1001	1010	0	10010000001100110011	ALTO
93	10010	1001	1010	0	000010000000001110001	BAJO
94	10010	1101	1100	0	000010000000001110001	ALTO
95	10010	1101	1100	0	000010000000001110001	BAJO
96	10011	1001	1100	0	00010000000000000000	ALTO
97	10011	1001	1100	0	00010000000000000000	BAJO
98	10100	1111	0000	0	01010000000000000000	ALTO
99	10100	1111	0000	0	01010000000000000000	BAJO
100	10101	0000	0000	0	00100000000000000000	ALTO
101	10101	0000	0000	0	00100000000000000000	BAJO
102	10110	0000	0000	0	00000000000000000000	ALTO
103	10110	0000	0000	0	00000000000000000000	BAJO
104	11000	0000	0000	0	00000000000000000000	ALTO
105	11000	0000	0000	0	00000000000000000000	BAJO

6. Imagenes de simulaciones

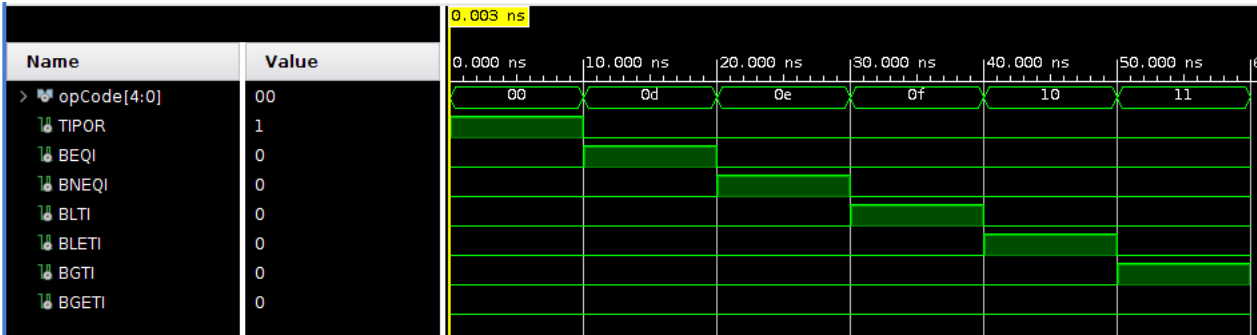
6.1. Registro



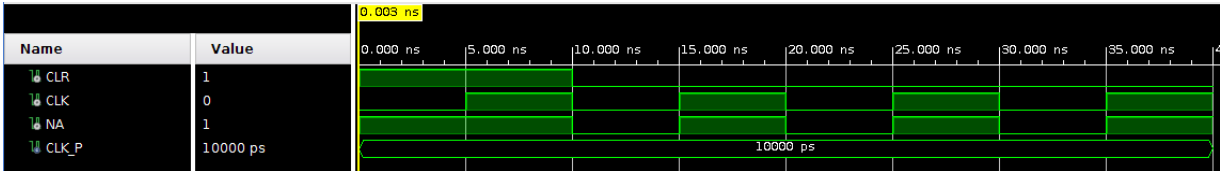
6.2. Bloque de condición



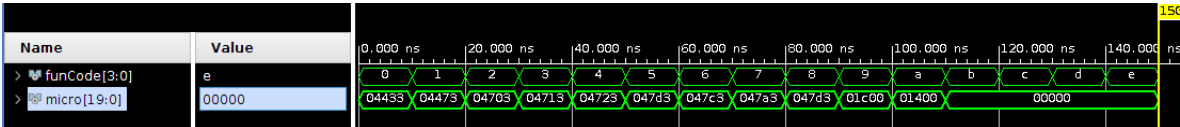
6.3. Bloque decodificador



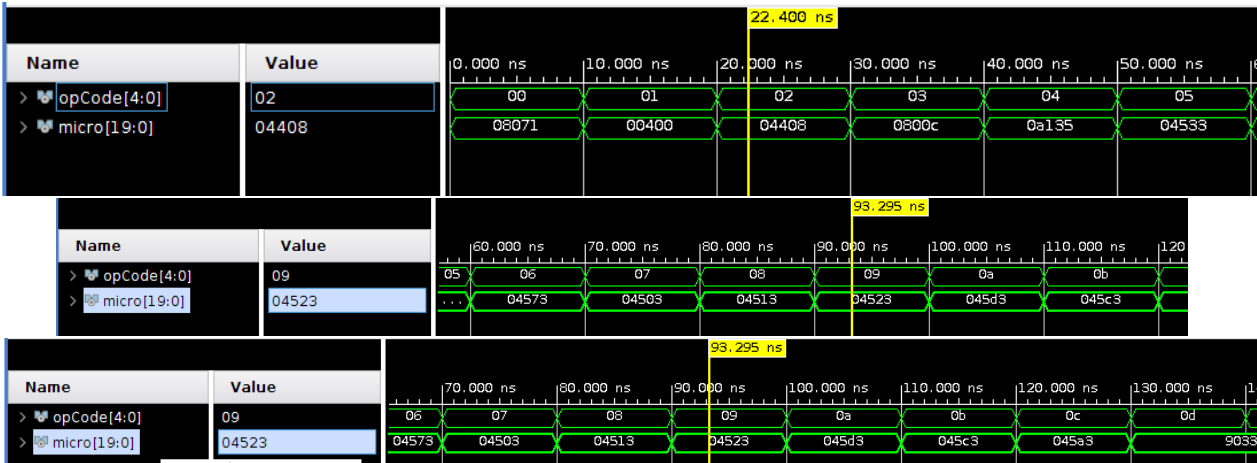
6.4. Bloque de nivel



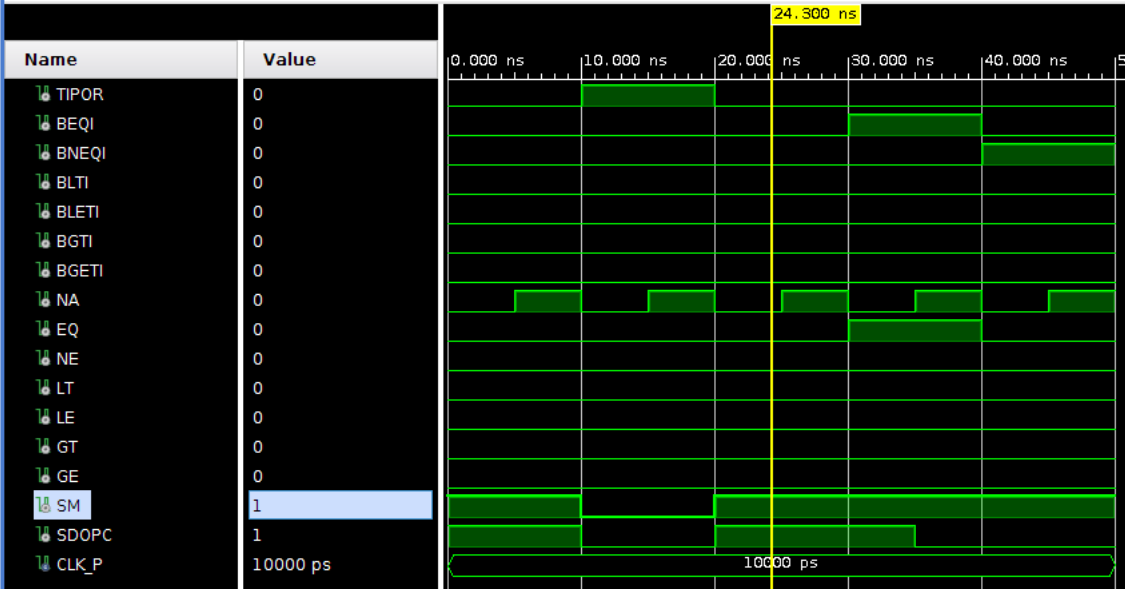
6.5. MFunCode



6.6. MOpCode



6.7. Unidad de Control

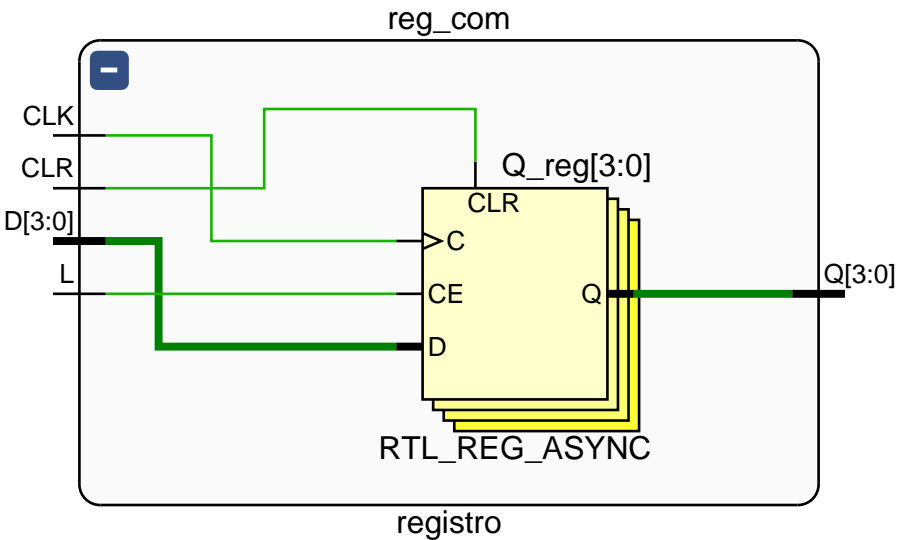


6.8. Arquitectura completa

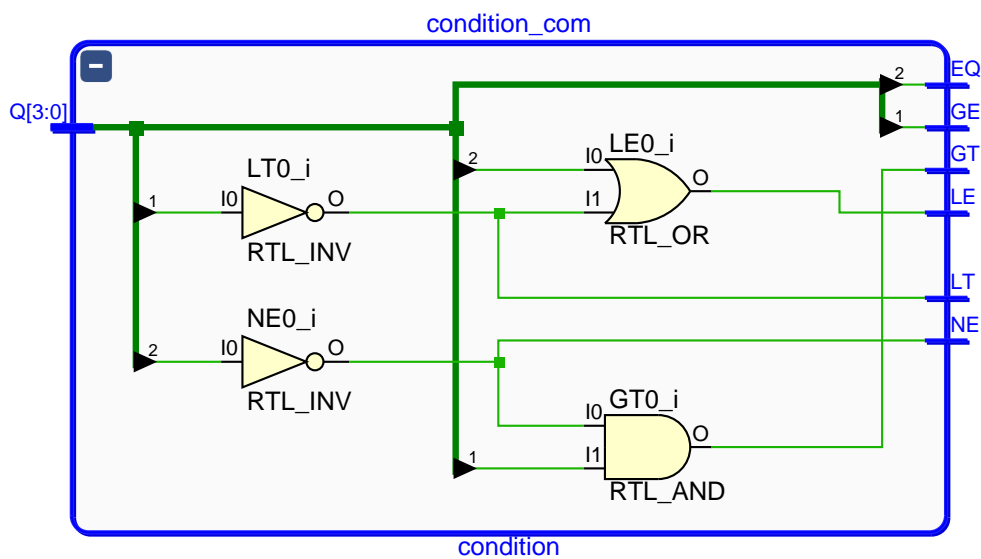


7. Digramas RTL

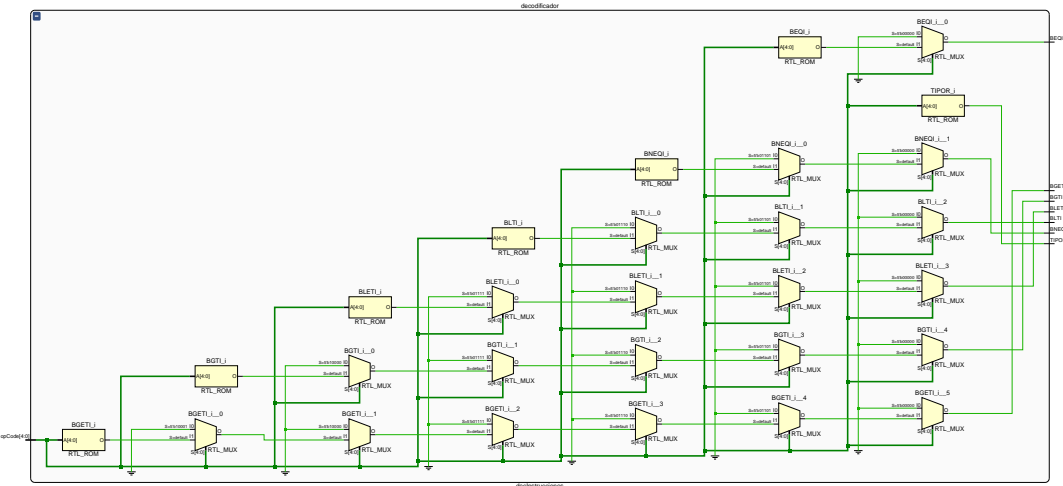
7.1. Registro



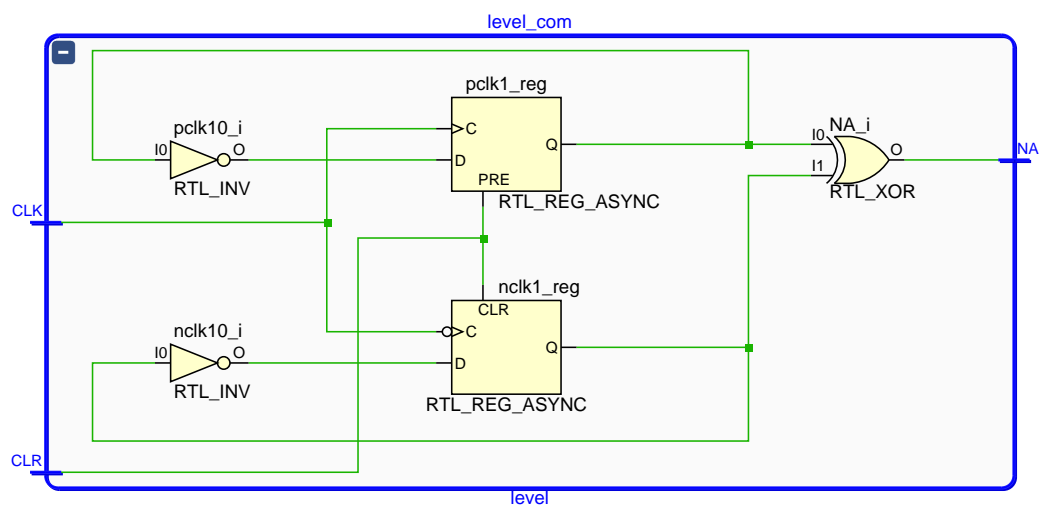
7.2. Bloque de condición



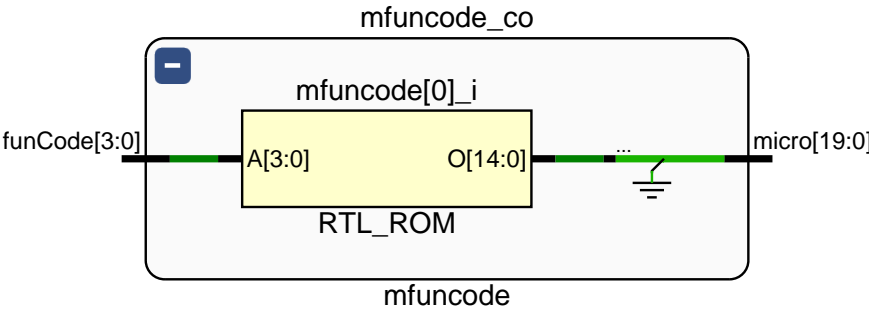
7.3. Bloque decodificador



7.4.    Bloque de nivel

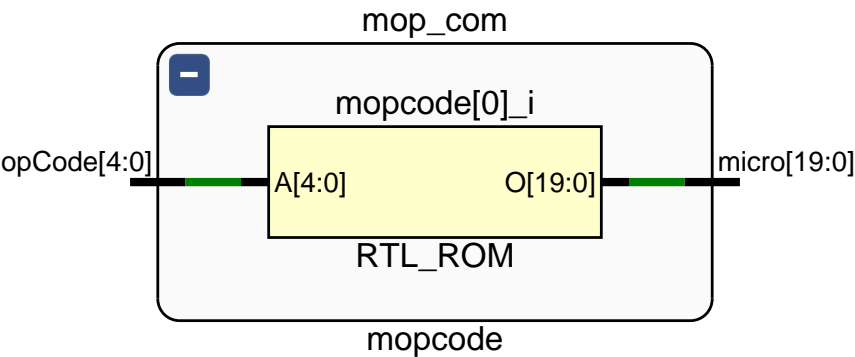


7.5. MFunCode

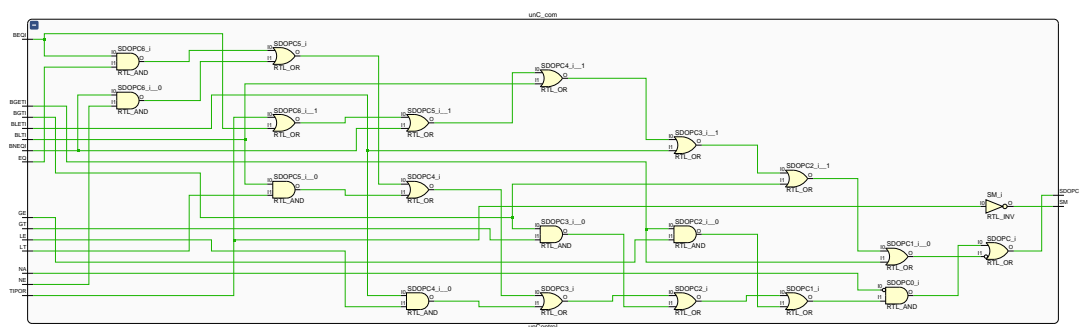




7.6. MOpCode



7.7. Unidad de Control



7.8. Arquitectura completa

