Reporte de practica 14

González Pardo Adrian

Junio 2020

1. Código fuente de los componentes:

1.1. Registro

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

entity registro is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           Q : out STD_LOGIC_VECTOR (3 downto 0);
           CLR,CLK,L : in STD_LOGIC);
end registro;

architecture Behavioral of registro is
begin
    process(CLR,CLK,L)
    begin
        if CLR='1' then
            Q<=(others=>'0');
        elsif rising_edge(CLK) then
            if L='1' then
                Q<=D;
            end if;
        end if;
    end process;
end Behavioral;
```

1.2. Bloque de condición

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

entity condition is
    Port ( Q : in STD_LOGIC_VECTOR (3 downto 0);
           EQ,NE,LT,LE,GT,GE : out STD_LOGIC);
end condition;

architecture Behavioral of condition is
begin
    --  Banderas
    --  Q(3) = OV
    --  Q(2) = Z
    --  Q(1) = C
    --  Q(0) = N
    process(Q)
        begin
        EQ<=Q(2);
        NE<=not(Q(2));
        LT<=not(Q(1));
        LE<=(Q(2)or(not(Q(1))));
```

```
        GT<=((not(Q(2)))and(Q(1)));
        GE<=Q(1);
    end process;


end Behavioral;
```

1.3. Bloque decodificador

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


entity decInstrucciones is
    Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
           TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI : out STD_LOGIC);
end decInstrucciones;

architecture Behavioral of decInstrucciones is

begin
    process(opCode)
    begin
        TIPOR<='0';
        BEQI<='0';
        BNEQI<='0';
        BLTI<='0';
        BLETI<='0';
        BGTI<='0';
        BGETI<='0';
        if opCode="00000" then --TIPOR
            TIPOR<='1';
        elsif opCode="01101" then --BEQI
            BEQI<='1';
        elsif opCode="01110" then --BNEQI
            BNEQI<='1';
        elsif opCode="01111" then --BLTI
            BLTI<='1';
        elsif opCode="10000" then --BLETI
            BLETI<='1';
        elsif opCode="10001" then --BGTI
            BGTI<='1';
        elsif opCode="10010" then --BGETI
            BGETI<='1';
        end if;
    end process;
end Behavioral;
```

1.4. Bloque de nivel

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

entity level is
    Port ( CLR,CLK : in STD_LOGIC;
           NA : out STD_LOGIC);
end level;

architecture Behavioral of level is
    signal pclk1,nclk1:std_logic;
begin
    process(CLR,CLK)
    begin
        if CLR='1' then
```

```
            pclk1<='1';
        elsif rising_edge(CLK) then
            pclk1<=not pclk1;
        end if;
    end process;
    process(CLK,CLR)
    begin
        if CLR='1' then
            nclk1<='0';
        elsif falling_edge(CLK) then
            nclk1<=not nclk1;
        end if;
    end process;
    NA<=pclk1 xor nclk1;
end Behavioral;
```

1.5. MFunCode

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

entity mfuncode is
    Port ( funCode : in STD_LOGIC_VECTOR (3 downto 0);
           micro : out STD_LOGIC_VECTOR (19 downto 0));
end mfuncode;

architecture Behavioral of mfuncode is
type mem is array(0 to 15) of std_logic_vector(19 downto 0);
constant mfuncode: mem:=("00000100010000110011",
                         "00000100010001110011",
                         "00000100011100000011",
                         "00000100011100010011",
                         "00000100011100100011",
                         "00000100011111010011",
                         "00000100011111000011",
                         "00000100011110100011",
                         "00000100011111010011",
                         "00000001110000000000",
                         "00000001010000000000",
                         others=>X"00000");
begin
    micro<=mfuncode(conv_integer(funcode));
end Behavioral;
```

1.6. MOpCode

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

entity mopcode is
    Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
           micro : out STD_LOGIC_VECTOR (19 downto 0));
end mopcode;

architecture Behavioral of mopcode is
type mem is array(0 to 31) of std_logic_vector(19 downto 0);
constant mopcode: mem:=("00001000000001110001",-- 0 VERIFI
                        "00000000010000000000",-- 1 LI
                        "00001000010000001000",-- 2 LWI
                        "00001000000000001100",-- 3 SWI
                        "00001010000100110101",-- 4 SW
                        "00001000010100110011",-- 5 ADDI
```

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

1.7. Unidad de Control

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

entity unControl is
    Port ( TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI,NA,EQ,NE,LT,LE,GT,GE : in STD_LOGIC;
           SM,SDOPC : out STD_LOGIC);
end unControl;

architecture Behavioral of unControl is
begin
    SDOPC<=(((not NA)and
        ((BEQI and EQ)or(BNEQI and NE)or(BLTI  and LT)or(BLETI and LE)or
        (BGTI  and GT)or(BGETI and GE)))or
      (not(TIPOR or BEQI or BNEQI or BLTI or BLETI or BGTI or BGETI)));
   SM<=not(TIPOR);
end Behavioral;
```

2. Código fuente del empaquetado

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

package paqueteHardware is

    component condition is
    Port ( Q : in STD_LOGIC_VECTOR (3 downto 0);
           EQ,NE,LT,LE,GT,GE : out STD_LOGIC);
    end component;

    component decInstrucciones is
    Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
           TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI : out STD_LOGIC);
    end component;

    component level is
    Port ( CLR,CLK : in STD_LOGIC;
           NA : out STD_LOGIC);
    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:
        std_logic;
18      signal Q:std_logic_vector(3 downto 0);
19  begin
20
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(
```

```
        funCode=>funCode,
        micro=>outMFunCode
    );

    decodificador:decInstrucciones port map(
        opCode=>opCode,
        TIPOR=>TIPOR,
        BEQI=>BEQI,
        BNEQI=>BNEQI,
        BLTI=>BLTI,
        BLETI=>BLETI,
        BGTI=>BGTI,
        BGETI=>BGETI
    );

    level_com:level port map(
        CLR=>CLR,
        CLK=>CLK,
        NA=>NA
    );

    reg_com:registro port map(
        D=>banderas,
        Q=>Q,
        CLR=>CLR,
        CLK=>CLK,
        L=>LF
    );

    condition_com:condition port map(
        Q=>Q,
        EQ=>EQ,
        NE=>NE,
        LT=>LT,
        LE=>LE,
        GT=>GT,
        GE=>GE
    );

    mop_com:mopcode port map(
        opCode=>muxMopCode,
        micro=>outMOpCode
    );

    unC_com:unControl port map(
        TIPOR=>TIPOR,
        BEQI=>BEQI,
        BNEQI=>BNEQI,
        BLTI=>BLTI,
        BLETI=>BLETI,
        BGTI=>BGTI,
        BGETI=>BGETI,
        NA=>NA,
        EQ=>EQ,
        NE=>NE,
        LT=>LT,
        LE=>LE,
        GT=>GT,
        GE=>GE,
        SM=>SM,
        SDOPC=>SDOPC
    );

end Behavioral;
```

4. Test-Bench:

4.1. Registro

```
library IEEE;
library WORK;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.paqueteHardware.registro;

entity registro_TB is
end registro_TB;

architecture Behavioral of registro_TB is
    signal CLK,CLR,L:std_logic:='0';
    signal D,Q:std_logic_vector(3 downto 0):=(others=>'0');
    constant CLK_P:time:=10ns;
begin
    reg_comp:registro port map(
        D=>D,
        Q=>Q,
        CLR=>CLR,
        CLK=>CLK,
        L=>L
    );
    clk_Pr:process
    begin
        wait for CLK_P/2;
        CLK<='1';
        wait for CLK_P/2;
        CLK<='0';
    end process;

    regTB:process
    begin
        CLR<='1';
        wait for CLK_P;
        CLR<='0';
        D<="0100";
        wait for CLK_P;
        L<='1';
        wait for CLK_P;
        D<="1100";
        wait for CLK_P;
        L<='0';
        D<="0101";
        wait for CLK_P;
        wait;
    end process;
end Behavioral;
```

4.2. Bloque de condición

```
library IEEE;
library WORK;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.paqueteHardware.condition;

entity condition_TB is
end condition_TB;

architecture Behavioral of condition_TB is
    signal Q:STD_LOGIC_VECTOR (3 downto 0);
    signal EQ,NE,LT,LE,GT,GE:STD_LOGIC;
begin
    condition_com:condition port map(
```

```
        Q=>Q,
        EQ=>EQ,
        NE=>NE,
        LT=>LT,
        LE=>LE,
        GT=>GT,
        GE=>GE
    );
    condition_TB:process
    begin
        Q<="0000";
        wait for 10 ns;
        Q<="0001";
        wait for 10 ns;
        Q<="0010";
        wait for 10 ns;
        Q<="0011";
        wait for 10 ns;
        Q<="0100";
        wait for 10 ns;
        Q<="0101";
        wait for 10 ns;
        Q<="0110";
        wait;
    end process;


end Behavioral;
```

4.3. Bloque decodificador

```
library IEEE;
library WORK;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.paqueteHardware.decInstrucciones;

entity decodificador_TB is
end decodificador_TB;

architecture Behavioral of decodificador_TB is
    signal opCode:STD_LOGIC_VECTOR (4 downto 0):=(others =>'0');
    signal TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI:STD_LOGIC:='0';
begin
    decodificador_com:decInstrucciones port map(
        opCode=>opCode,
        TIPOR=>TIPOR,
        BEQI=>BEQI,
        BNEQI=>BNEQI,
        BLTI=>BLTI,
        BLETI=>BLETI,
        BGTI=>BGTI,
        BGETI=>BGETI
    );

    decInstrucciones_TB:process
    begin
        opCode <= "00000";
        wait for 10 ns;
        opCode <= "01101";
        wait for 10 ns;
        opCode <= "01110";
        wait for 10 ns;
        opCode <= "01111";
        wait for 10 ns;
        opCode <= "10000";
        wait for 10 ns;
```

```
        opCode <= "10001";
        wait for 10 ns;
        opCode <= "10010";
        wait;
    end process;

end Behavioral;
```

4.4. Bloque de nivel

```
library IEEE;
library WORK;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.paqueteHardware.level;

entity level_TB is
end level_TB;

architecture Behavioral of level_TB is
    signal CLR,CLK,NA:std_logic:='0';
    constant CLK_P:time:=10ns;
begin
    level_com:level port map(
        CLR=>CLR,
        CLK=>CLK,
        NA=>NA
    );

    clk_Pr:process
    begin
        wait for CLK_P/2;
        CLK<='1';
        wait for CLK_P/2;
        CLK<='0';
    end process;

    levelTB:process
    begin
        CLR <= '1';
        wait for CLK_P;
        CLR <= '0';
        wait for CLK_P;
        wait for CLK_P;
        wait for CLK_P;
        wait;
    end process;


end Behavioral;
```

4.5. MFunCode

```
library IEEE;
library WORK;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.paqueteHardware.mfuncode;

entity mfuncode_TB is
end mfuncode_TB;

architecture Behavioral of mfuncode_TB is
    signal funCode:STD_LOGIC_VECTOR (3 downto 0):=(others=>'0');
    signal micro:STD_LOGIC_VECTOR (19 downto 0):=(others=>'0');
begin
```

```
    mfuncode_com:mfuncode port map(
        funCode=>funCode,
        micro=>micro
    );

    mfuncode_TB:process
    begin
        funCode <= "0000";
        wait for 10 ns;
        funCode <= "0001";
        wait for 10 ns;
        funCode <= "0010";
        wait for 10 ns;
        funCode <= "0011";
        wait for 10 ns;
        funCode <= "0100";
        wait for 10 ns;
        funCode <= "0101";
        wait for 10 ns;
        funCode <= "0110";
        wait for 10 ns;
        funCode <= "0111";
        wait for 10 ns;
        funCode <= "1000";
        wait for 10 ns;
        funCode <= "1001";
        wait for 10 ns;
        funCode <= "1010";
        wait for 10 ns;
        funCode <= "1011";
        wait for 10 ns;
        funCode <= "1100";
        wait for 10 ns;
        funCode <= "1101";
        wait for 10 ns;
        funCode <= "1110";
        wait for 10 ns;
        funCode <= "1111";
        wait;
    end process;


end Behavioral;
```

4.6. MOpCode

```
library IEEE;
library WORK;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.paqueteHardware.mopcode;

entity mopcode_TB is
end mopcode_TB;

architecture Behavioral of mopcode_TB is
    signal opCode:STD_LOGIC_VECTOR (4 downto 0):=(others=>'0');
    signal micro:STD_LOGIC_VECTOR (19 downto 0):=(others=>'0');
begin
    mopcode_com:mopcode port map(
        opCode=>opCode,
        micro=>micro
    );
    mopcode_TB:process
    begin
        opCode <= "00000";
        wait for 10 ns;
```

```
        opCode <= "00001";
        wait for 10 ns;
        opCode <= "00010";
        wait for 10 ns;
        opCode <= "00011";
        wait for 10 ns;
        opCode <= "00100";
        wait for 10 ns;
        opCode <= "00101";
        wait for 10 ns;
        opCode <= "00110";
        wait for 10 ns;
        opCode <= "00111";
        wait for 10 ns;
        opCode <= "01000";
        wait for 10 ns;
        opCode <= "01001";
        wait for 10 ns;
        opCode <= "01010";
        wait for 10 ns;
        opCode <= "01011";
        wait for 10 ns;
        opCode <= "01100";
        wait for 10 ns;
        opCode <= "01101";
        wait for 10 ns;
        opCode <= "01110";
        wait for 10 ns;
        opCode <= "01111";
        wait;
    end process;
end Behavioral;
```

4.7. Unidad de Control

```
library IEEE;
library WORK;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.paqueteHardware.unControl;

entity unControl_TB is
end unControl_TB;

architecture Behavioral of unControl_TB is
    signal TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI,NA,EQ,NE,LT,LE,GT,GE,SM,SDOPC:STD_LOGIC
    :='0';
    constant CLK_P:time:=10ns;
begin
    unControl_com:unControl port map(
        TIPOR=>TIPOR,
        BEQI=>BEQI,
        BNEQI=>BNEQI,
        BLTI=>BLTI,
        BLETI=>BLETI,
        BGTI=>BGTI,
        BGETI=>BGETI,
        NA=>NA,EQ=>EQ,
        NE=>NE,
        LT=>LT,
        LE=>LE,
        GT=>GT,
        GE=>GE,
        SM=>SM,
        SDOPC=>SDOPC
    );
    clk_Pr:process
```

```
    begin
        wait for CLK_P/2;
        NA<='1';
        wait for CLK_P/2;
        NA<='0';
    end process;

    unControl_TB:process
    begin
        TIPOR<='0';
        BEQI<='0';
        BNEQI<='0';
        BLTI<='0';
        BLETI<='0';
        BGTI<='0';
        BGETI<='0';
        EQ<='0';
        NE<='0';
        LT<='0';
        LE<='0';
        GT<='0';
        GE<='0';
        wait for CLK_P;
        TIPOR<='1';
        wait for CLK_P;
        TIPOR<='0';
        wait for CLK_P;
        BEQI<='1';
        EQ<='1';
        wait for CLK_P;
        BEQI<='0';
        EQ<='0';
        BNEQI<='1';
        NE<='0';
        wait for CLK_P;
        wait;
    end process;
end Behavioral;
```

4.8. Arquitectura completa

```
library IEEE;
LIBRARY STD;
use IEEE.STD_LOGIC_1164.ALL;
USE STD.TEXTIO.ALL;
USE ieee.std_logic_TEXTIO.ALL;  --PERMITE USAR STD_LOGIC
USE ieee.std_logic_UNSIGNED.ALL;
USE ieee.std_logic_ARITH.ALL;


entity arquitectura_TB is
end arquitectura_TB;

architecture Behavioral of arquitectura_TB is
    component arquitectura is
    Port ( funCode,banderas : in STD_LOGIC_VECTOR (3 downto 0);
           CLK,CLR : in STD_LOGIC;
           opCode : in STD_LOGIC_VECTOR (4 downto 0);
           microinstruccion : out STD_LOGIC_VECTOR (19 downto 0));
    end component;
    signal funCode,banderas : STD_LOGIC_VECTOR (3 downto 0):=(others=>'0');
    signal CLK,CLR : STD_LOGIC:='0';
    signal opCode : STD_LOGIC_VECTOR (4 downto 0):=(others=>'0');
    signal microinstruccion : STD_LOGIC_VECTOR (19 downto 0):=(others=>'0');
    constant CLK_P : time:=10 ns;
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
90              read(LINEA_ENT,VAR_Flags);
91              banderas <= VAR_Flags;
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
```

```
00000 1000 0010 0 1
00000 1001 0000 0 0
00000 1010 0000 0 0
00000 1011 0000 0 0
00000 1100 0000 0 0
00001 0111 0000 0 0
00010 0100 0000 0 0
00011 1000 0000 0 0
00100 0110 0000 0 0
00101 0000 0010 0 1
00110 0110 0001 0 1
00111 0100 0011 0 1
01000 1010 0100 0 1
01001 0100 1000 0 1
01010 0001 1100 0 1
01011 0011 0101 0 1
01100 1111 1010 0 1
10111 0000 0000 0 1
01101 1111 0000 0 1
01101 1011 0010 0 1
01101 1101 0010 0 1
01110 1110 0010 0 1
01110 1100 0000 0 1
01110 0011 0000 0 1
01111 0001 1100 0 1
01111 0000 1000 0 1
01111 0010 0100 0 1
10000 0100 0000 0 1
10000 0110 1110 0 1
10000 0101 1000 0 1
10001 0111 1010 0 1
10001 1010 1100 0 1
10001 1000 0000 0 1
10010 1111 1000 0 1
10010 1001 1010 0 1
10010 1101 1100 0 1
10011 1001 1100 0 0
10100 1111 0000 0 0
10101 0000 0000 0 0
10110 0000 0000 0 0
11000 0000 0000 0 0
```

5.2. Salida

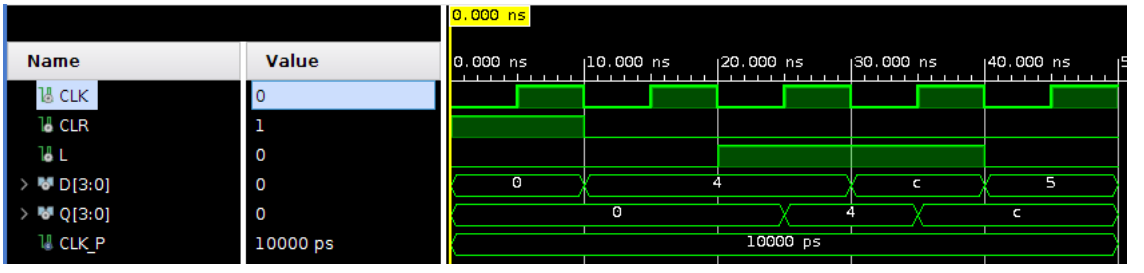
```
 _OP_CODE_ _FUN_CODE _BANDERAS  __CLR___ _MICRONST             __NIVEL_
00000       0000       0000          1       00000100010000110011   ALTO
00000       0000       0000          1       00000100010000110011   BAJO
00000       0000       0000          1       00000100010000110011   ALTO
00000       0000       0000          1       00000100010000110011   BAJO
00000       0000       0001          0       00000100010000110011   ALTO
00000       0000       0001          0       00000100010000110011   BAJO
00000       0000       0010          0       00000100010000110011   ALTO
00000       0000       0010          0       00000100010000110011   BAJO
00000       0001       0001          0       00000100010001110011   ALTO
00000       0001       0001          0       00000100010001110011   BAJO
00000       0010       0100          0       00000100011100000011   ALTO
00000       0010       0100          0       00000100011100000011   BAJO
00000       0011       1100          0       00000100011100010011   ALTO
00000       0011       1100          0       00000100011100010011   BAJO
00000       0100       0011          0       00000100011100100011   ALTO
00000       0100       0011          0       00000100011100100011   BAJO
00000       0101       1000          0       00000100011111010011   ALTO
00000       0101       1000          0       00000100011111010011   BAJO
00000       0110       0001          0       00000100011111000011   ALTO
00000       0110       0001          0       00000100011111000011   BAJO
00000       0111       0100          0       00000100011110100011   ALTO
```

00000	0111	0100	0	00000100011110100011	BAJO
00000	1000	0010	0	00000100011111010011	ALTO
00000	1000	0010	0	00000100011111010011	BAJO
00000	1001	0000	0	00000001110000000000	ALTO
00000	1001	0000	0	00000001110000000000	BAJO
00000	1010	0000	0	00000001010000000000	ALTO
00000	1010	0000	0	00000001010000000000	BAJO
00000	1011	0000	0	00000000000000000000	ALTO
00000	1011	0000	0	00000000000000000000	BAJO
00000	1100	0000	0	00000000000000000000	ALTO
00000	1100	0000	0	00000000000000000000	BAJO
00001	0111	0000	0	00000000010000000000	ALTO
00001	0111	0000	0	00000000010000000000	BAJO
00010	0100	0000	0	00000100010000001000	ALTO
00010	0100	0000	0	00000100010000001000	BAJO
00011	1000	0000	0	00001000000000001100	ALTO
00011	1000	0000	0	00001000000000001100	BAJO
00100	0110	0000	0	00001010000100110101	ALTO
00100	0110	0000	0	00001010000100110101	BAJO
00101	0000	0010	0	00000100010100110011	ALTO
00101	0000	0010	0	00000100010100110011	BAJO
00110	0110	0001	0	00000100010101110011	ALTO
00110	0110	0001	0	00000100010101110011	BAJO
00111	0100	0011	0	00000100010100000011	ALTO
00111	0100	0011	0	00000100010100000011	BAJO
01000	1010	0100	0	00000100010100010011	ALTO
01000	1010	0100	0	00000100010100010011	BAJO
01001	0100	1000	0	00000100010100100011	ALTO
01001	0100	1000	0	00000100010100100011	BAJO
01010	0001	1100	0	00000100010111010011	ALTO
01010	0001	1100	0	00000100010111010011	BAJO
01011	0011	0101	0	00000100010111000011	ALTO
01011	0011	0101	0	00000100010111000011	BAJO
01100	1111	1010	0	00000100010110100011	ALTO
01100	1111	1010	0	00000100010110100011	BAJO
10111	0000	0000	0	00001110010100110001	ALTO
10111	0000	0000	0	00001110010100110001	BAJO
01101	1111	0000	0	00001000000001110001	ALTO
01101	1111	0000	0	00001000000001110001	BAJO
01101	1011	0010	0	00001000000001110001	ALTO
01101	1011	0010	0	00001000000001110001	BAJO
01101	1101	0010	0	00001000000001110001	ALTO
01101	1101	0010	0	00001000000001110001	BAJO
01110	1110	0010	0	10010000001100110011	ALTO
01110	1110	0010	0	00001000000001110001	BAJO
01110	1100	0000	0	10010000001100110011	ALTO
01110	1100	0000	0	00001000000001110001	BAJO
01110	0011	0000	0	10010000001100110011	ALTO
01110	0011	0000	0	00001000000001110001	BAJO
01111	0001	1100	0	10010000001100110011	ALTO
01111	0001	1100	0	00001000000001110001	BAJO
01111	0000	1000	0	10010000001100110011	ALTO
01111	0000	1000	0	00001000000001110001	BAJO
01111	0010	0100	0	10010000001100110011	ALTO
01111	0010	0100	0	00001000000001110001	BAJO
10000	0100	0000	0	10010000001100110011	ALTO
10000	0100	0000	0	00001000000001110001	BAJO
10000	0110	1110	0	10010000001100110011	ALTO
10000	0110	1110	0	00001000000001110001	BAJO
10000	0101	1000	0	10010000001100110011	ALTO
10000	0101	1000	0	00001000000001110001	BAJO
10001	0111	1010	0	10010000001100110011	ALTO
10001	0111	1010	0	00001000000001110001	BAJO
10001	1010	1100	0	00001000000001110001	ALTO
10001	1010	1100	0	00001000000001110001	BAJO
10001	1000	0000	0	00001000000001110001	ALTO
10001	1000	0000	0	00001000000001110001	BAJO

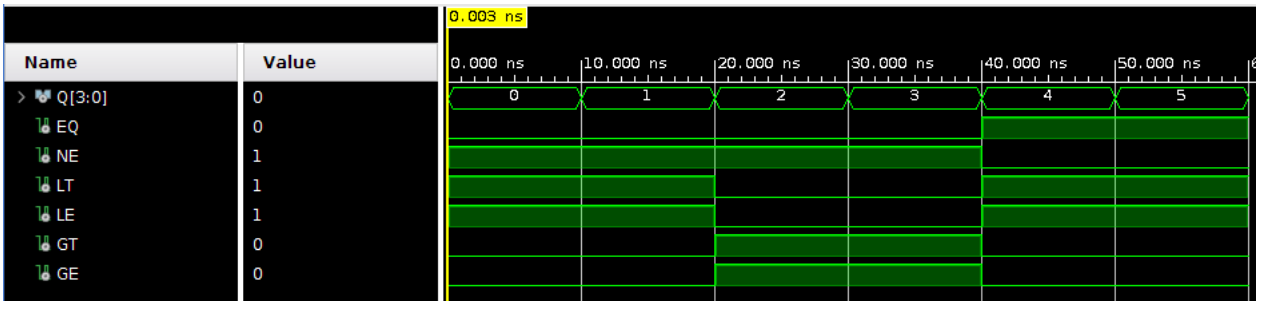
```
10010       1111        1000        0       00001000000001110001    ALTO
10010       1111        1000        0       00001000000001110001    BAJO
10010       1001        1010        0       10010000001100110011    ALTO
10010       1001        1010        0       00001000000001110001    BAJO
10010       1101        1100        0       00001000000001110001    ALTO
10010       1101        1100        0       00001000000001110001    BAJO
10011       1001        1100        0       00010000000000000000    ALTO
10011       1001        1100        0       00010000000000000000    BAJO
10100       1111        0000        0       01010000000000000000    ALTO
10100       1111        0000        0       01010000000000000000    BAJO
10101       0000        0000        0       00100000000000000000    ALTO
10101       0000        0000        0       00100000000000000000    BAJO
10110       0000        0000        0       00000000000000000000    ALTO
10110       0000        0000        0       00000000000000000000    BAJO
11000       0000        0000        0       00000000000000000000    ALTO
11000       0000        0000        0       00000000000000000000    BAJO
```

6. Imagenes de simulaciones

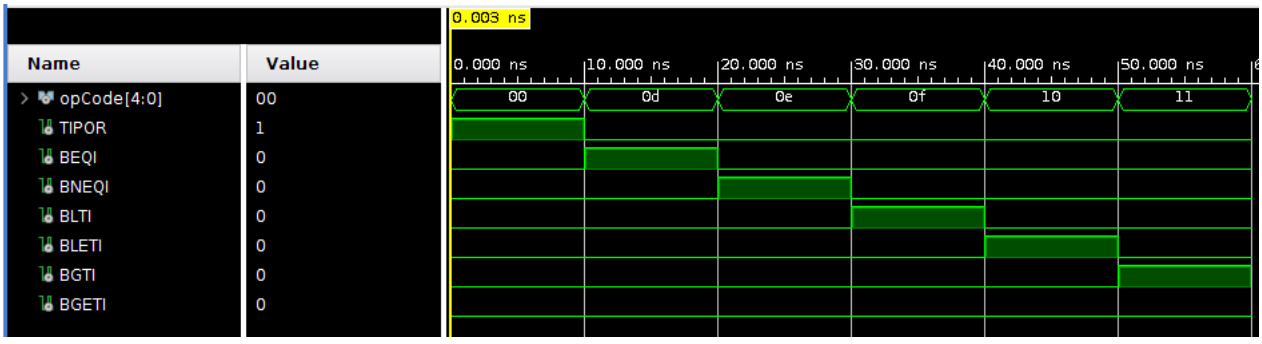
6.1. Registro



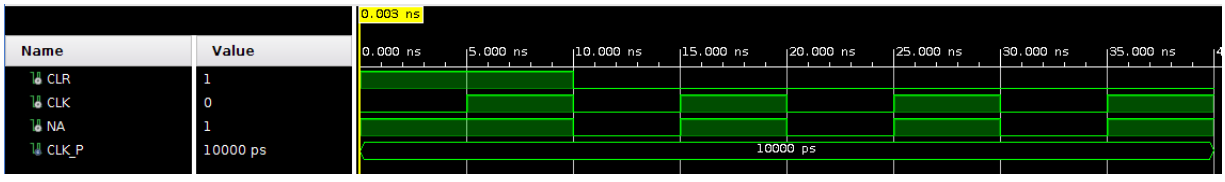
6.2. Bloque de condición



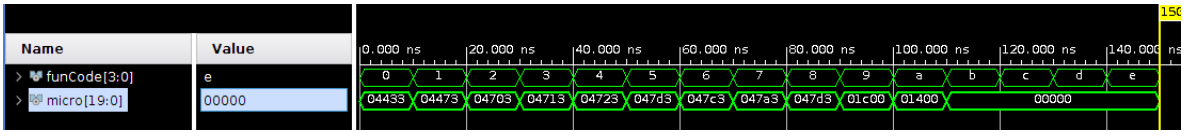
6.3. Bloque decodificador



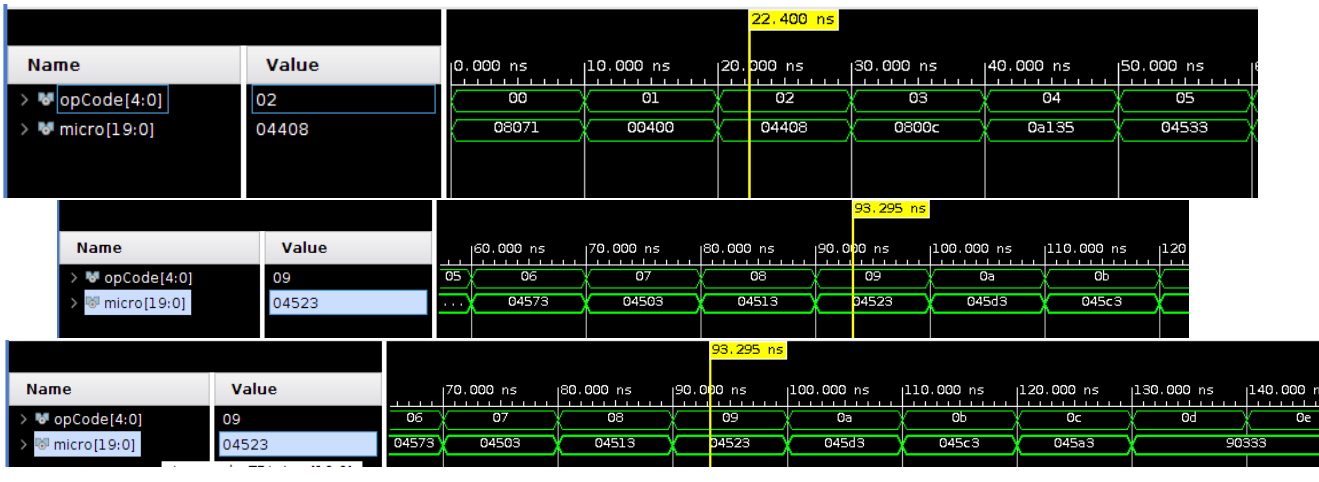
6.4. Bloque de nivel



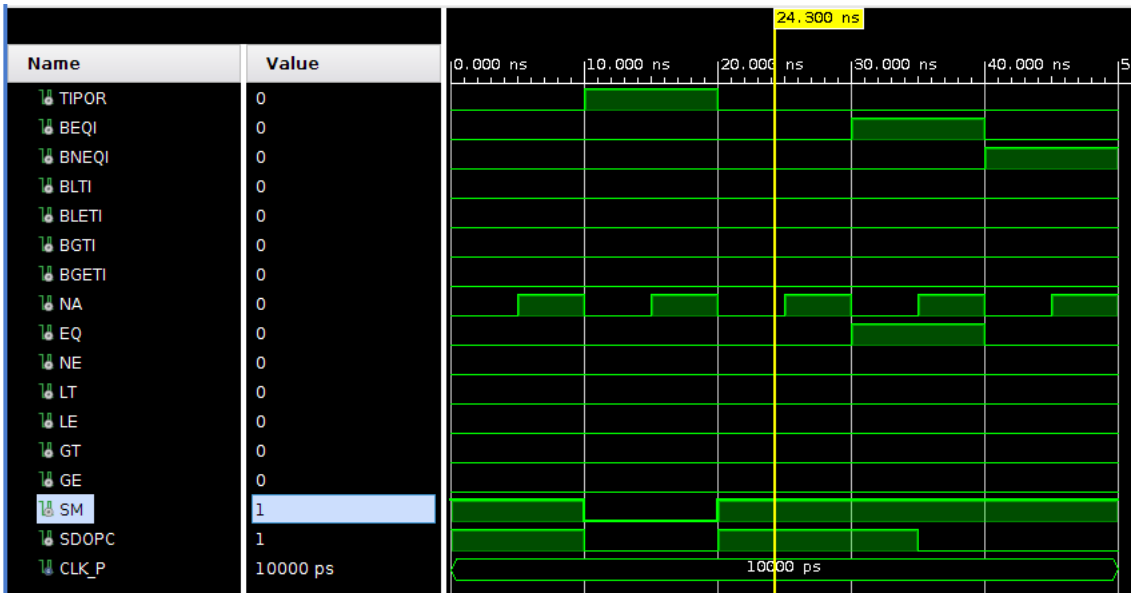
6.5. MFunCode



6.6. MOpCode



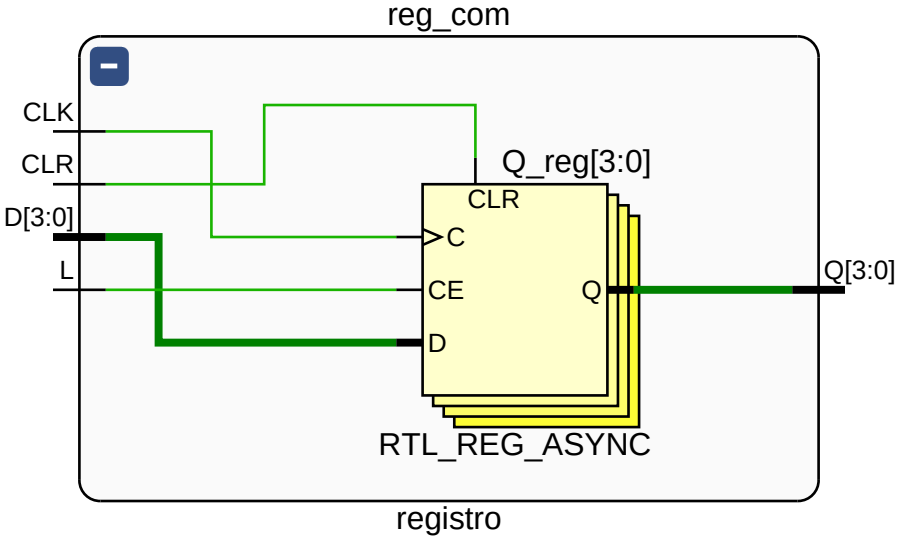
6.7. Unidad de Control



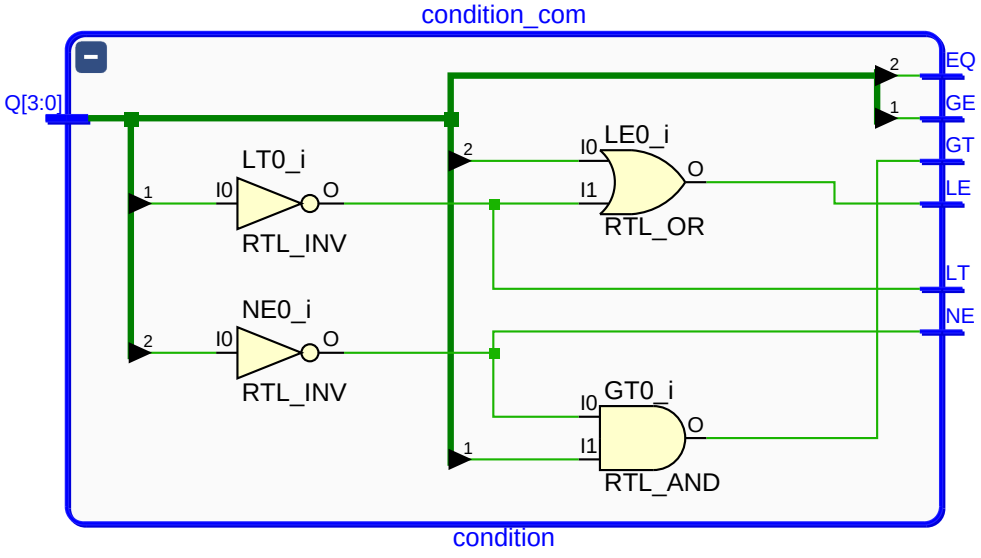
6.8. Arquitectura completa

7. Digramas RTL

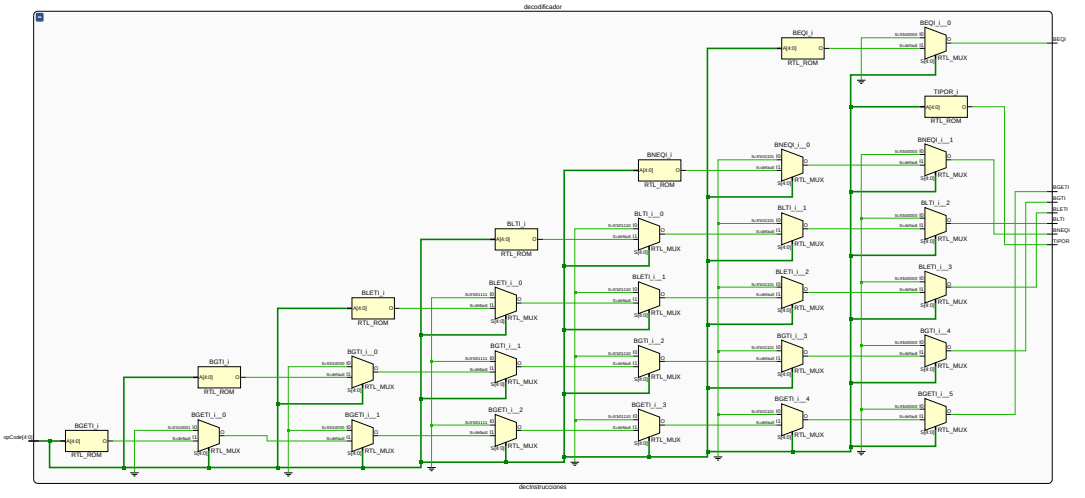
7.1. Registro



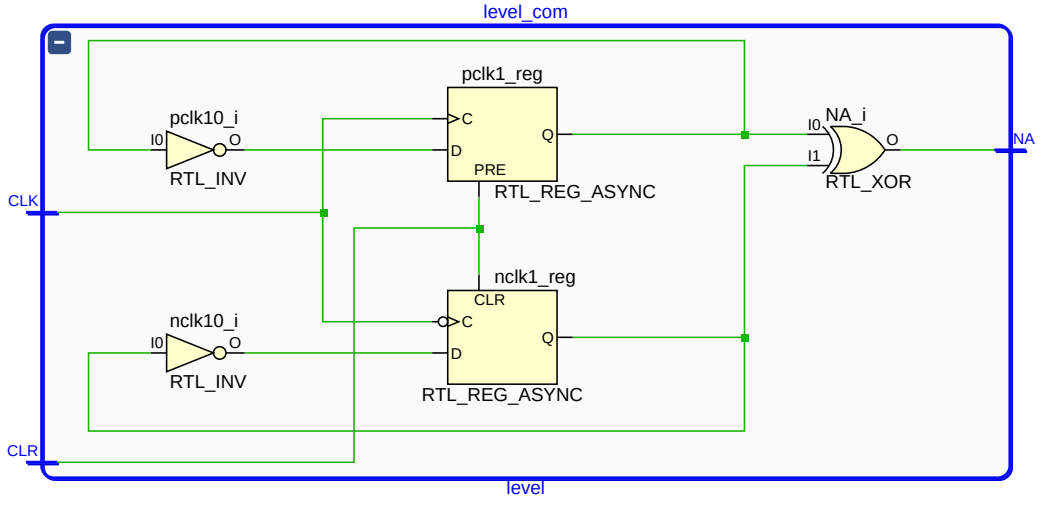
7.2. Bloque de condición



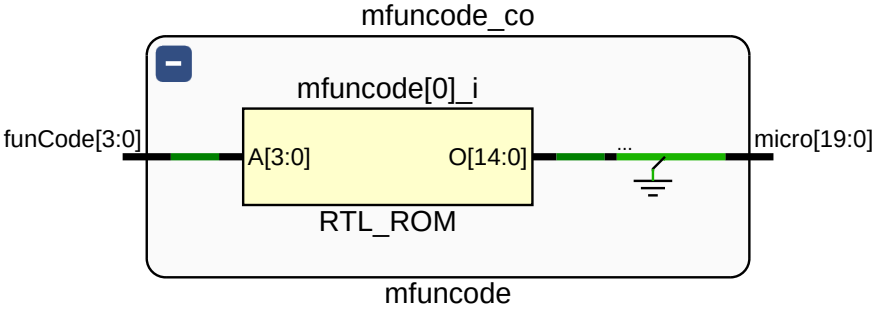
7.3. Bloque decodificador



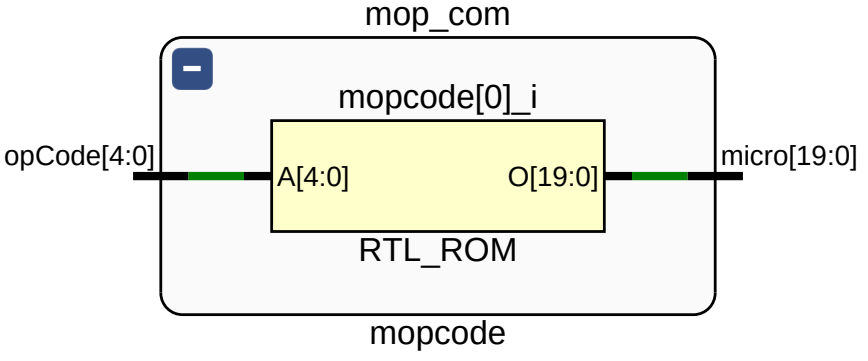
7.4. Bloque de nivel



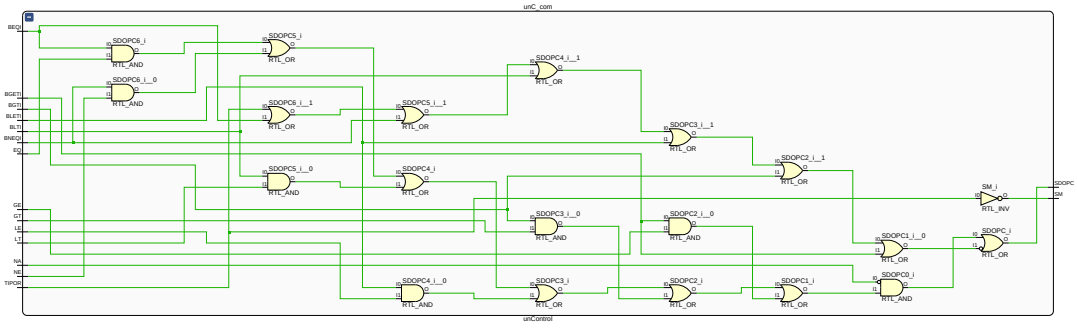
7.5. MFunCode


mfuncode_co
mfuncode[0]_i
funCode[3:0]
A[3:0]
O[14:0]
...
micro[19:0]
RTL_ROM
mfuncode

7.6. MOpCode


mop_com
mopcode[0]_i
opCode[4:0]
A[4:0]
O[19:0]
micro[19:0]
RTL_ROM
mopcode

7.7. Unidad de Control



7.8. Arquitectura completa