Martínez Coronel Brayan Yosafat

# Código de Implementación

## Decodificador

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity decodificador is

Port (

id : in STD\_LOGIC\_VECTOR (4 downto 0);

TIPOR, BEQI, BNEI, BLTI, BLETI, BGTI, BGETI : out STD\_LOGIC

);

end decodificador;

architecture Behavioral of decodificador is

begin

with id select

TIPOR <= '1' when "00000",

'0' when others;

with id select

BEQI <= '1' when "01101",

'0' when others;

with id select

BNEI <= '1' when "01110",

'0' when others;

with id select

BLTI <= '1' when "01111",

'0' when others;

with id select

BLETI <= '1' when "10000",

'0' when others;

with id select

BGTI <= '1' when "10001",

'0' when others;

with id select

BGETI <= '1' when "10010",

'0' when others;

end Behavioral;

## Memoria de Código de Función

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity memoria\_fun\_code is

generic (

m : integer := 4;

n : integer := 20

);

Port ( index : in STD\_LOGIC\_VECTOR (m-1 downto 0);

code : out STD\_LOGIC\_VECTOR (n-1 downto 0)

);

end entity;

architecture Behavioral of memoria\_fun\_code is

type arreglo is array (0 to (2\*\*m - 1)) of STD\_LOGIC\_VECTOR (n-1 downto 0);

signal mem : arreglo := (others=>(others=>'0'));

begin

mem(0) <= "00000100010000110011"; --ADD

mem(1) <= "00000100010001110011"; --SUB

mem(2) <= "00000100010000000011"; --AND

mem(3) <= "00000100010000010011"; --OR

mem(4) <= "00000100010000100011"; --XOR

mem(5) <= "00000100010011010011"; --NAND

mem(6) <= "00000100010011000011"; --NOR

mem(7) <= "00000100010001100011"; --XNOR

mem(8) <= "00000100010011000011"; --NOT

mem(9) <= "00000001110000000000"; --SLL

mem(10) <= "00000001010000000000"; --SRL

code <= mem(conv\_integer(index));

end Behavioral;

## Memoria de código de Operación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity memoria\_op\_code is

generic (

m : integer := 5;

n : integer := 20

);

Port ( index : in STD\_LOGIC\_VECTOR (m-1 downto 0);

code : out STD\_LOGIC\_VECTOR (n-1 downto 0)

);

end entity;

architecture Behavioral of memoria\_op\_code is

type arreglo is array (0 to (2\*\*m - 1)) of STD\_LOGIC\_VECTOR (n-1 downto 0);

signal mem : arreglo := (others=>(others=>'0'));

begin

mem(0) <= "00001000000001110001"; --Verificacion

mem(1) <= "00000000010000000000"; --LI

mem(2) <= "00000100010000001000"; --LWI

mem(3) <= "00001000000000001100"; --SWI

mem(4) <= "00001010000100110101"; --SW

mem(5) <= "00000100010100110011"; --ADDI

mem(6) <= "00000100010101110011"; --SUBI

mem(7) <= "00000110010100000011"; --ANDI

mem(8) <= "00000110010100010011"; --ORI

mem(9) <= "00000100010100100011"; --XORI

mem(10) <= "00000100010111010011"; --NANDI

mem(11) <= "00000100010111000011"; --NORI

mem(12) <= "00000100010101100011"; --XNORI

mem(13) <= "10010000001100110011"; --BEQI

mem(14) <= "10010000001100110011"; --BNEI

mem(15) <= "10010000001100110011"; --BLTI

mem(16) <= "10010000001100110011"; --BLETI

mem(17) <= "10010000001100110011"; --BGTI

mem(18) <= "10010000001100110011"; --BGETI

mem(19) <= "00010000000000000000"; --B

mem(20) <= "01010000000000000000"; --CALL

mem(21) <= "00100000000000000000"; --RET

mem(22) <= "00000000000000000000"; --NOP

mem(23) <= "00000110010100110001"; --LW

code <= mem(conv\_integer(index));

end Behavioral;

## Nivel

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity nivel is

Port (

CLK, CLR : in STD\_LOGIC;

NA : out STD\_LOGIC

);

end nivel;

architecture Behavioral of nivel is

signal up\_clk, down\_clk : STD\_LOGIC := '0';

begin

alto: process (CLK, CLR)

begin

if (CLR = '1') then

up\_clk <= '0';

elsif (rising\_edge(CLK)) then

up\_clk <= not up\_clk;

end if;

end process;

bajo: process (CLK, CLR, up\_clk)

begin

if (CLR = '1') then

down\_clk <= '0';

elsif (falling\_edge(CLK)) then

down\_clk <= not down\_clk;

end if;

end process;

NA <= up\_clk xor down\_clk;

end Behavioral;

## Condición

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity condiciones is

Port (

Q : in STD\_LOGIC\_VECTOR (3 downto 0);

EQ, NE, LT, LE, GT, GE : out STD\_LOGIC

);

end condiciones;

architecture Behavioral of condiciones is

begin

-- Q = "OV N Z C"

EQ <= Q(1);

NE <= not Q(1);

LT <= not Q(0);

LE <= Q(1) or not Q(0);

GT <= not Q(1) and Q(0);

GE <= Q(0);

end Behavioral;

## Registro de Banderas

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity banderas is

Port (

LF, CLK, CLR : in STD\_LOGIC;

D : in STD\_LOGIC\_VECTOR (3 downto 0);

Q : out STD\_LOGIC\_VECTOR (3 downto 0)

);

end banderas;

architecture Behavioral of banderas is

signal reg : STD\_LOGIC\_VECTOR (3 downto 0);

begin

process (CLK, CLR, D)

begin

if (CLR = '1') then

reg <= "0000";

elsif falling\_edge(CLK) then

if (LF = '1') then

reg <= D;

else

reg <= "0000";

end if;

end if;

end process;

Q <= reg;

end Behavioral;

## Control

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity control is

Port (

CLK, CLR : in STD\_LOGIC;

EQ, NE, LT, LE, GT, GE, NA : in STD\_LOGIC;

TIPOR, BEQI, BNEI, BLTI, BLETI, BGTI, BGETI : in STD\_LOGIC;

SDOPC, SM : out STD\_LOGIC

);

end control;

architecture Behavioral of control is

type estado is (E0);

signal actual, sig : estado;

begin

--Para el registro del estado

process (CLK, CLR)

begin

if CLR = '1' then

actual <= E0;

elsif rising\_edge(CLK) then

actual <= sig;

end if;

end process;

process (actual, NA)

begin

SDOPC <= '0';

SM <= '1';

case actual is

when E0 =>

if TIPOR = '1' then

SM <= '0';

elsif BEQI = '1' then

if NA = '0' then

SDOPC <= '0';

else

if EQ = '1' then

SDOPC <= '1';

else

SDOPC <= '0';

end if;

end if;

elsif BNEI = '1' then

if NA = '0' then

SDOPC <= '0';

else

if NE = '1' then

SDOPC <= '1';

else

SDOPC <= '0';

end if;

end if;

elsif BLTI = '1' then

if NA = '0' then

SDOPC <= '0';

else

if LT = '1' then

SDOPC <= '1';

else

SDOPC <= '0';

end if;

end if;

elsif BLETI = '1' then

if NA = '0' then

SDOPC <= '0';

else

if LE = '1' then

SDOPC <= '1';

else

SDOPC <= '0';

end if;

end if;

elsif BGTI = '1' then

if NA = '0' then

SDOPC <= '0';

else

if GT = '1' then

SDOPC <= '1';

else

SDOPC <= '0';

end if;

end if;

elsif BGETI = '1' then

if NA = '0' then

SDOPC <= '0';

else

if GE = '1' then

SDOPC <= '1';

else

SDOPC <= '0';

end if;

end if;

else

SDOPC <= '1';

end if;

end case;

sig <= E0;

end process;

end Behavioral;

## Unidad de Control

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use WORK.PAQUETE\_COMPONENTES.ALL;

entity unidad\_control is

Port (

CLK, CLR : in STD\_LOGIC;

D, sufix : in STD\_LOGIC\_VECTOR (3 downto 0);

prefix : in STD\_LOGIC\_VECTOR (4 downto 0);

microinst : out STD\_LOGIC\_VECTOR (19 downto 0)

);

end unidad\_control;

architecture Behavioral of unidad\_control is

signal index\_op : STD\_LOGIC\_VECTOR (4 downto 0);

signal Q : STD\_LOGIC\_VECTOR (3 downto 0);

signal out\_op, out\_fun, output : STD\_LOGIC\_VECTOR (19 downto 0);

signal s\_TIPOR, s\_BEQI, s\_BNEI, s\_BLTI, s\_BLETI, s\_BGTI, s\_BGETI : STD\_LOGIC;

signal s\_EQ, s\_NE, s\_LT, s\_LE, s\_GT, s\_GE, SDOPC, SM, NA : STD\_LOGIC;

begin

t\_mem\_op : memoria\_op\_code Port map (

index => index\_op,

code => out\_op

);

t\_mem\_fun : memoria\_fun\_code Port map (

index => sufix,

code => out\_fun

);

t\_deco : decodificador Port map (

id => prefix,

TIPOR => s\_TIPOR,

BEQI => s\_BEQI,

BNEI => s\_BNEI,

BLTI => s\_BLTI,

BLETI => s\_BLETI,

BGTI => s\_BGTI,

BGETI => s\_BGETI

);

t\_banderas : banderas Port map (

LF => output(0),

CLK => CLK,

CLR => CLR,

D => D,

Q => Q

);

t\_condiciones : condiciones Port map (

Q => Q,

EQ => s\_EQ,

NE => s\_NE,

LT => s\_LT,

LE => s\_LE,

GT => s\_GT,

GE => s\_GE

);

t\_control : control Port map (

CLK => CLK,

CLR => CLR,

EQ => s\_EQ,

NE => s\_NE,

LT => s\_LT,

LE => s\_LE,

GT => s\_GT,

GE => s\_GE,

NA => NA,

TIPOR => s\_TIPOR,

BEQI => s\_BEQI,

BNEI => s\_BNEI,

BLTI => s\_BLTI,

BLETI => s\_BLETI,

BGTI => s\_BGTI,

BGETI => s\_BGETI,

SDOPC => SDOPC,

SM => SM

);

t\_nivel : nivel Port map (

CLK => CLK,

CLR => CLR,

NA => NA

);

with SDOPC select

index\_op <= "00000" when '0',

prefix when others;

with SM select

output <= out\_fun when '0',

out\_op when others;

microinst <= output;

end Behavioral;

# Código de simulación

## Decodificador

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_deco is

-- Port ( );

end tb\_deco;

architecture Behavioral of tb\_deco is

component decodificador is

Port (

id : in STD\_LOGIC\_VECTOR (4 downto 0);

TIPOR, BEQI, BNEI, BLTI, BLETI, BGTI, BGETI : out STD\_LOGIC

);

end component;

signal opCode : STD\_LOGIC\_VECTOR (4 downto 0);

signal TIPOR, BEQI, BNEQI, BLTI, BLETI, BGTI, BGETI : STD\_LOGIC;

begin

deco : decodificador Port map (

id => opCode,

TIPOR => TIPOR,

BEQI => BEQI,

BNEI => BNEQI,

BLTI => BLTI,

BLETI => BLETI,

BGTI => BGTI,

BGETI => BGETI

);

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;

## Memoria de Código de Función

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_fun\_code is

-- Port ( );

end tb\_fun\_code;

architecture Behavioral of tb\_fun\_code is

component memoria\_fun\_code is

generic (

m : integer := 4;

n : integer := 20

);

Port ( index : in STD\_LOGIC\_VECTOR (m-1 downto 0);

code : out STD\_LOGIC\_VECTOR (n-1 downto 0)

);

end component;

signal funCode : STD\_LOGIC\_VECTOR (3 downto 0);

signal microFuncion : STD\_LOGIC\_VECTOR (19 downto 0);

begin

fun : memoria\_fun\_code Port map (

index => funCode,

code => microFuncion

);

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;

end process;

end Behavioral;

## Memoria de código de Operación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_op\_code is

-- Port ( );

end tb\_op\_code;

architecture Behavioral of tb\_op\_code is

component memoria\_op\_code is

generic (

m : integer := 5;

n : integer := 20

);

Port ( index : in STD\_LOGIC\_VECTOR (m-1 downto 0);

code : out STD\_LOGIC\_VECTOR (n-1 downto 0)

);

end component;

signal opCode : STD\_LOGIC\_VECTOR (4 downto 0);

signal microOpCode : STD\_LOGIC\_VECTOR (19 downto 0);

begin

op : memoria\_op\_code Port map (

index => OpCode,

code => microOpCode

);

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;

end process;

end Behavioral;

## Nivel

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_nivel is

-- Port ( );

end tb\_nivel;

architecture Behavioral of tb\_nivel is

component nivel is

Port (

CLK, CLR : in STD\_LOGIC;

NA : out STD\_LOGIC

);

end component;

signal clr, clk, na : STD\_LOGIC;

begin

niv : nivel Port map (

clk => clk,

clr => clr,

na => na

);

reloj : process begin

clk <= '0';

wait for 5 ns;

clk <= '1';

wait for 5 ns;

end process;

process

begin

clr <= '1';

wait for 3 ns;

clr <= '0';

wait;

end process;

end Behavioral;

## Condición

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_condiciones is

-- Port ( );

end tb\_condiciones;

architecture Behavioral of tb\_condiciones is

component condiciones is

Port (

Q : in STD\_LOGIC\_VECTOR (3 downto 0);

EQ, NE, LT, LE, GT, GE : out STD\_LOGIC

);

end component;

signal EQ, NE, LT, LE, GT, GE : STD\_LOGIC;

signal Q : STD\_LOGIC\_VECTOR (3 downto 0);

begin

con : condiciones Port map (

Q => Q,

EQ => EQ,

NE => NE,

LT => LT,

LE => LE,

GT => GT,

GE => GE

);

--Q[OV, N, Z, C]

process

begin

Q <= "0010";

wait for 10 ns;

Q <= "1101";

wait for 10 ns;

Q <= "0001";

wait for 10 ns;

Q <= "0000";

wait for 10 ns;

Q <= "1111";

wait;

end process;

end Behavioral;

## Registro de Banderas

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_banderas is

-- Port ( );

end tb\_banderas;

architecture Behavioral of tb\_banderas is

component banderas is

Port (

LF, CLK, CLR : in STD\_LOGIC;

D : in STD\_LOGIC\_VECTOR (3 downto 0);

Q : out STD\_LOGIC\_VECTOR (3 downto 0)

);

end component;

signal clr, clk, LF : STD\_LOGIC;

signal D, Q : STD\_LOGIC\_VECTOR (3 downto 0);

begin

reg : banderas port map(

D => D,

clr => clr,

clk => clk,

LF => LF,

Q => Q

);

reloj : process begin

clk <= '0';

wait for 5 ns;

clk <= '1';

wait for 5 ns;

end process;

process

begin

clr <= '1';

wait until falling\_edge(clk);

clr <= '0';

LF <= '1';

D <= "1111";

wait until falling\_edge(clk);

D <= "0000";

wait until falling\_edge(clk);

D <= "0001";

wait until falling\_edge(clk);

D <= "0010";

wait until falling\_edge(clk);

D <= "0011";

wait until falling\_edge(clk);

LF <= '0';

D <= "0000";

wait until falling\_edge(clk);

D <= "0001";

wait;

end process;

end Behavioral;

## Control

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_control is

-- Port ( );

end tb\_control;

architecture Behavioral of tb\_control is

component control is

Port (

CLK, CLR : in STD\_LOGIC;

EQ, NE, LT, LE, GT, GE, NA : in STD\_LOGIC;

TIPOR, BEQI, BNEI, BLTI, BLETI, BGTI, BGETI : in STD\_LOGIC;

SDOPC, SM : out STD\_LOGIC

);

end component;

signal clk, clr, EQ, NE, LT, LE, GT, GE, nivel, BGETI, BGTI, BLETI, BLTI, BNEQI, BEQI, TIPOR, SDOPC, SM : STD\_LOGIC := '0';

begin

uc : control Port map (

clk => clk,

clr => clr,

EQ => EQ,

NE => NE,

LT => LT,

LE => LE,

GT => GT,

GE => GE,

na => nivel,

BGETI => BGETI,

BGTI => BGTI,

BLETI => BLETI,

BLTI => BLTI,

BNEI => BNEQI,

BEQI => BEQI,

TIPOR => TIPOR,

SDOPC => SDOPC,

SM => SM

);

reloj : process begin

clk <= '0';

wait for 5 ns;

clk <= '1';

wait for 5 ns;

end process;

process

begin

clr <= '1';

wait until rising\_edge(clk);

clr <= '0';

TIPOR <= '1';

wait until rising\_edge(clk);

TIPOR <= '0';

LT <= '1';

BLTI <= '1';

wait until rising\_edge(clk);

LT <= '0';

wait until rising\_edge(clk);

BLTI <= '0';

LE <= '1';

BLETI <= '1';

wait until rising\_edge(clk);

LE <= '0';

wait until rising\_edge(clk);

clr <= '1';

wait;

end process;

end Behavioral;

## Unidad de Control

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use STD.TEXTIO.ALL;

use IEEE.std\_logic\_TEXTIO.ALL;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_unidad\_control is

-- Port ( );

end tb\_unidad\_control;

architecture Behavioral of tb\_unidad\_control is

constant clk\_period : time := 10 ns;

component unidad\_control is

Port (

CLK, CLR, LF : in STD\_LOGIC;

D, sufix : in STD\_LOGIC\_VECTOR (3 downto 0);

prefix : in STD\_LOGIC\_VECTOR (4 downto 0);

microinst : out STD\_LOGIC\_VECTOR (19 downto 0)

);

end component;

signal CLK, CLR, LF : STD\_LOGIC;

signal D, sufix : STD\_LOGIC\_VECTOR (3 downto 0);

signal prefix : STD\_LOGIC\_VECTOR (4 downto 0);

signal microinst : STD\_LOGIC\_VECTOR (19 downto 0);

begin

t\_unidad : unidad\_control port map (

CLK => CLK,

CLR => CLR,

LF =>LF,

D => D,

sufix => sufix,

prefix => prefix,

microinst => microinst

);

clk\_process : process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

sim\_proc: process

constant ALTO : string (1 to 4) := "ALTO";

constant BAJO : string (1 to 4) := "BAJO";

file ARCH\_SALIDA : TEXT;

variable LINEA\_SALIDA : line;

file ARCH\_ENTRADA : TEXT;

variable LINEA\_ENTRADA : line;

variable CADENA\_1 : string (1 to 8);

variable CADENA\_2 : string (1 to 20);

variable CADENA\_3 : string (1 to 5);

variable v\_CLK, v\_CLR, v\_LF : STD\_LOGIC;

variable v\_D, v\_sufix : STD\_LOGIC\_VECTOR (3 downto 0);

variable v\_prefix : STD\_LOGIC\_VECTOR (4 downto 0);

variable v\_microinst : STD\_LOGIC\_VECTOR (19 downto 0);

begin

file\_open(ARCH\_ENTRADA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica11\entradas.txt", READ\_MODE);

file\_open(ARCH\_SALIDA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica11\salidas.txt", WRITE\_MODE);

readline(ARCH\_ENTRADA, LINEA\_ENTRADA); -- salta la primera linea

CADENA\_1 := " OP\_CODE";

write(LINEA\_SALIDA, CADENA\_1, right, CADENA\_1'LENGTH + 1);

CADENA\_1 := "FUN\_CODE";

write(LINEA\_SALIDA, CADENA\_1, right, CADENA\_1'LENGTH + 1);

CADENA\_1 := "BANDERAS";

write(LINEA\_SALIDA, CADENA\_1, right, CADENA\_1'LENGTH + 1);

CADENA\_3 := " CLR";

write(LINEA\_SALIDA, CADENA\_3, right, CADENA\_3'LENGTH + 1);

CADENA\_3 := " LF";

write(LINEA\_SALIDA, CADENA\_3, right, CADENA\_3'LENGTH + 1);

CADENA\_2 := " MICROINSTRUCCION";

write(LINEA\_SALIDA, CADENA\_2, right, CADENA\_2'LENGTH + 1);

CADENA\_3 := "NIVEL";

write(LINEA\_SALIDA, CADENA\_3, right, CADENA\_3'LENGTH + 1);

writeline(ARCH\_SALIDA, LINEA\_SALIDA);-- escribe la linea en el archivo

FOR I IN 1 TO 52 LOOP

readline(ARCH\_ENTRADA, LINEA\_ENTRADA); -- lee una linea completa

read(LINEA\_ENTRADA, v\_prefix);

prefix <= v\_prefix;

read(LINEA\_ENTRADA, v\_sufix);

sufix <= v\_sufix;

read(LINEA\_ENTRADA, v\_D);

D <= v\_D;

read(LINEA\_ENTRADA, v\_CLR);

CLR <= v\_CLR;

read(LINEA\_ENTRADA, v\_LF);

LF <= v\_LF;

WAIT UNTIL RISING\_EDGE(CLK); --ESPERO AL FLANCO

v\_microinst := microinst;

write(LINEA\_SALIDA, v\_prefix, right, 8);

write(LINEA\_SALIDA, v\_sufix, right, 9);

write(LINEA\_SALIDA, v\_D, right, 9);

write(LINEA\_SALIDA, v\_CLR, right, 6);

write(LINEA\_SALIDA, v\_LF, right, 6);

write(LINEA\_SALIDA, v\_microinst, right, 22);

write(LINEA\_SALIDA, ALTO, right, 6);

writeline(ARCH\_SALIDA, LINEA\_SALIDA);

WAIT UNTIL FALLING\_EDGE(CLK);

v\_microinst := microinst;

write(LINEA\_SALIDA, v\_prefix, right, 8);

write(LINEA\_SALIDA, v\_sufix, right, 9);

write(LINEA\_SALIDA, v\_D, right, 9);

write(LINEA\_SALIDA, v\_CLR, right, 6);

write(LINEA\_SALIDA, v\_LF, right, 6);

write(LINEA\_SALIDA, v\_microinst, right, 22);

write(LINEA\_SALIDA, BAJO, right, 6);

writeline(ARCH\_SALIDA, LINEA\_SALIDA);

end loop;

file\_close(ARCH\_ENTRADA); -- cierra el archivo

file\_close(ARCH\_SALIDA); -- cierra el archivo

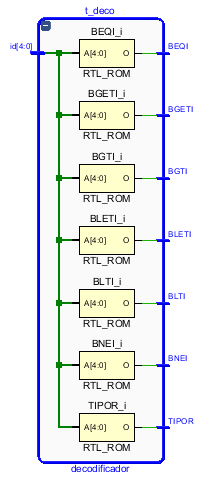
wait;

end process;

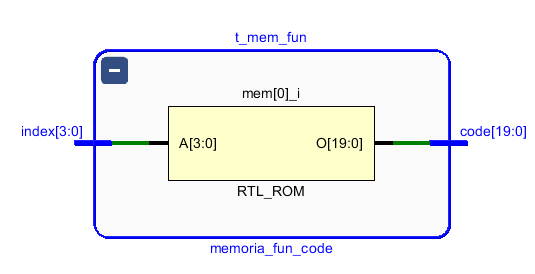
end Behavioral;

# Diagramas RTL

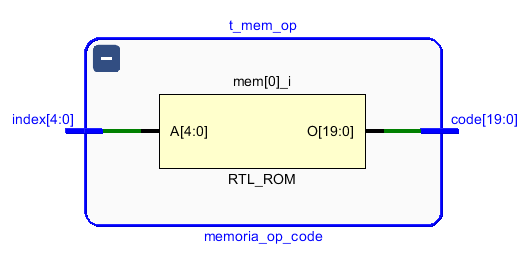
## Decodificador



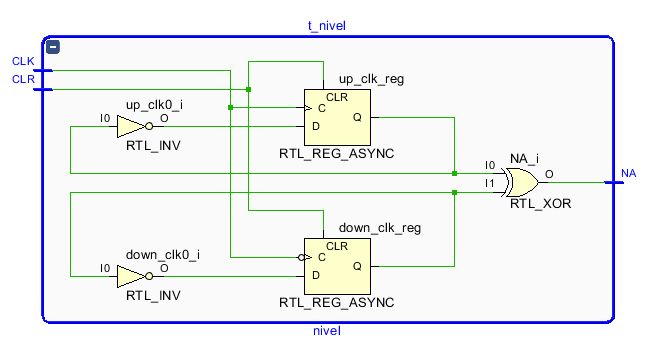
## Memoria de Código de Función



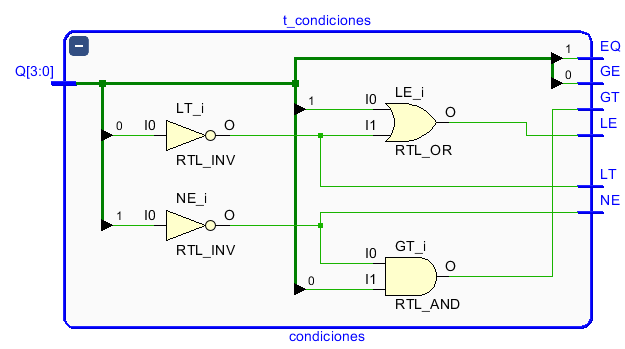
## Memoria de código de Operación



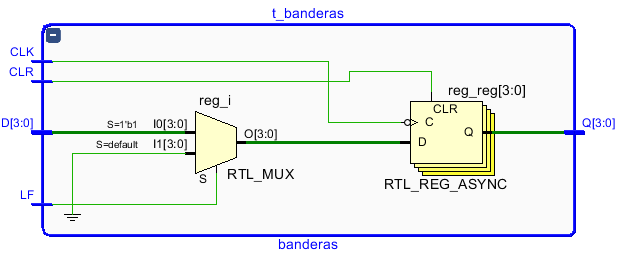
## Nivel



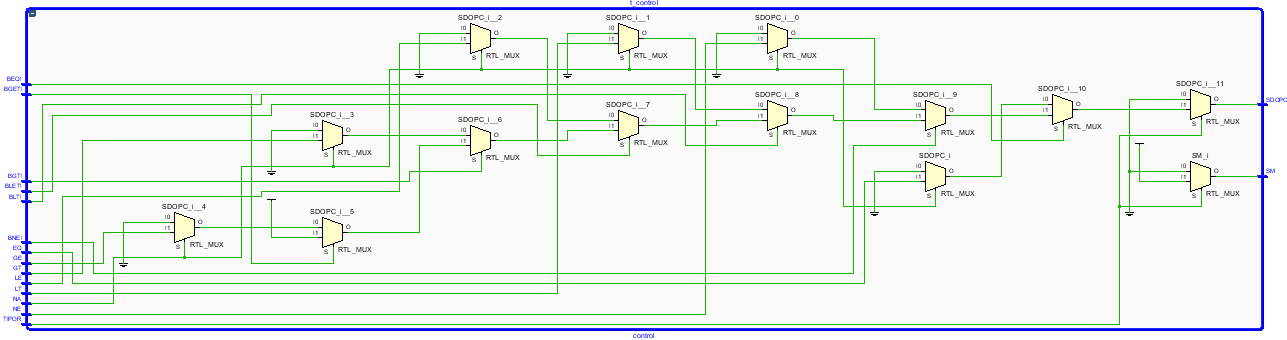
## Condición



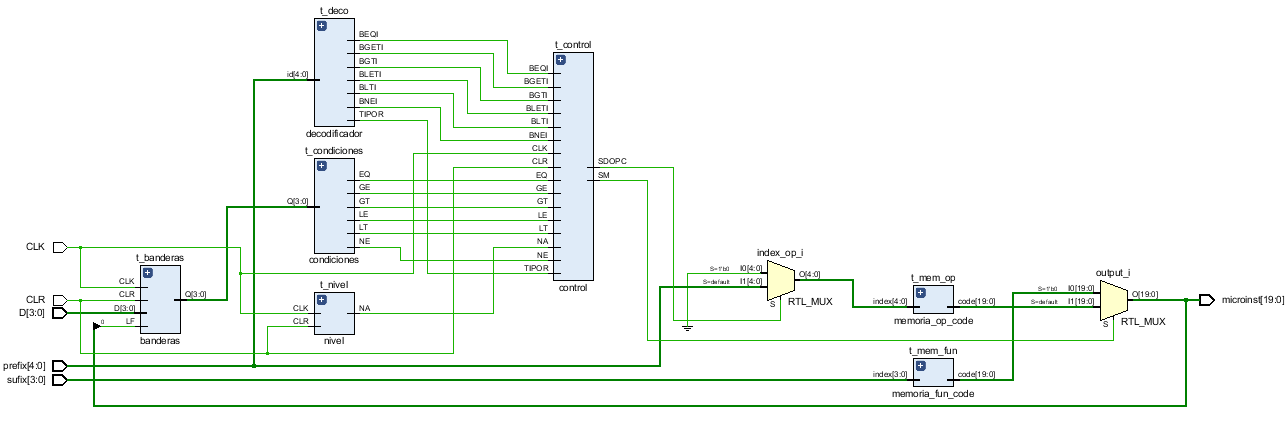
## Registro de Banderas



## Control

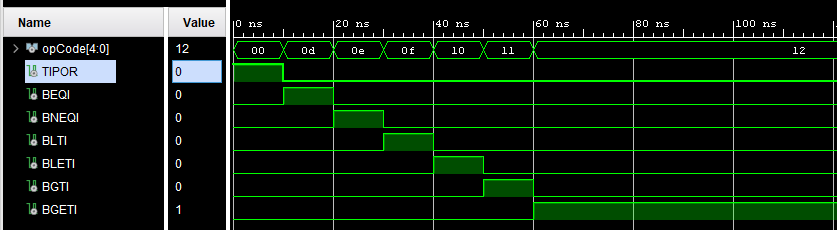


## Unidad de Control

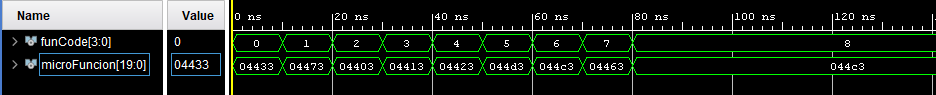


# Formas de onda

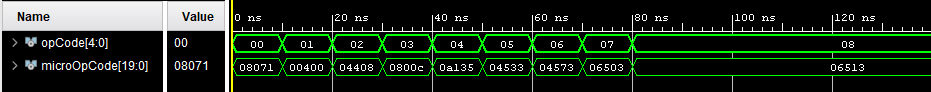
## Decodificador



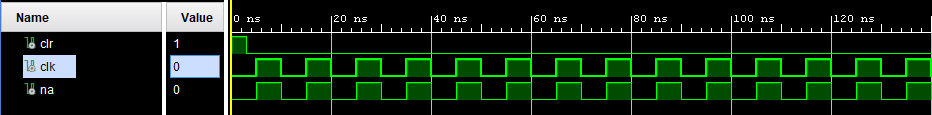
## Memoria de Código de Función



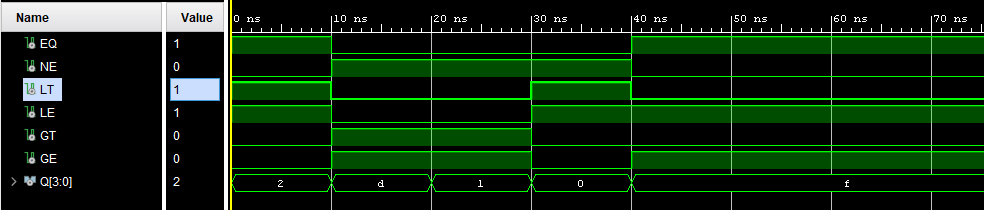
## Memoria de código de Operación



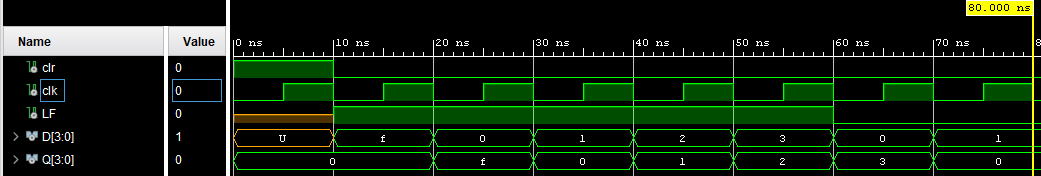
## Nivel



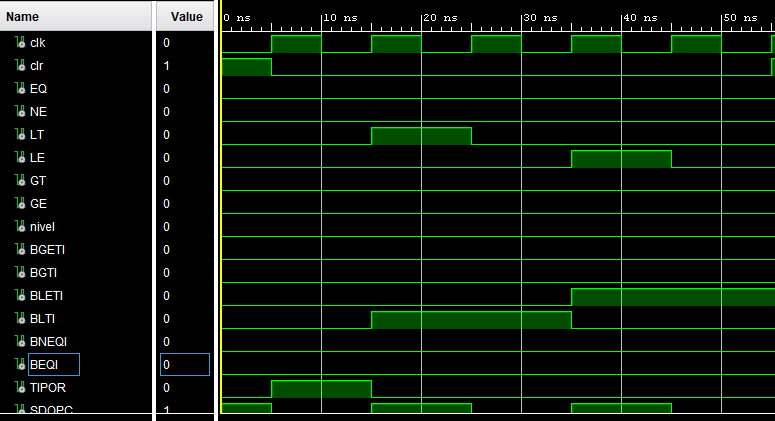
## Condición

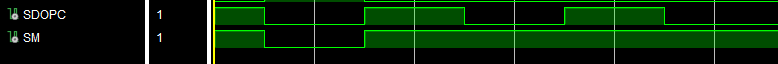


## Registro de Banderas



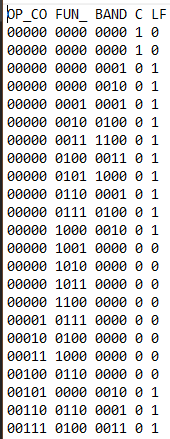
## Control

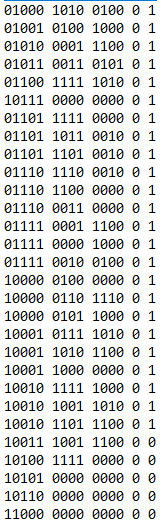




# Archivos

## Entrada





## Salida

