Instituto Politécnico Nacional

Escuela Superior de Cómputo

Arquitectura de Computadoras

Practica 14: Unidad de Control

Nombre: *Sampayo Hernández Mauro*

Grupo: *3CV8*

Profesor: *Nayeli Vega García*

Fecha de entrega: *28 de junio del 2020*

Elementos de la Arquitectura

Código de Implementación

- **MfunCode**

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

entity MfunCode is
     generic (
        D : integer := 20;
        A : integer := 4);
    Port ( funCode : in  STD_LOGIC_VECTOR (A-1 downto 0);
           salidaD : out  STD_LOGIC_VECTOR (D-1 downto 0));
end MfunCode;

architecture Behavioral of MfunCode is
    constant ADD: std_logic_vector := X"04433";
    constant SUB: std_logic_vector := X"04473";
    constant OpAND: std_logic_vector := X"04403";
    constant OpOR:  std_logic_vector := X"04413";
    constant OpXOR: std_logic_vector := X"04423";
    constant OpNAND: std_logic_vector := X"044d3";
    constant OpNOR: std_logic_vector := X"044c3";
    constant OpXNOR: std_logic_vector := X"044a3";
    constant OpNOT: std_logic_vector := X"044d3";
    constant OpSLL: std_logic_vector := X"01400";
    constant OpSRL: std_logic_vector := X"01c00";

    type banco is array (0 to (2**A)-1) of std_logic_vector(D-1 downto
0);
    constant memoria : banco := (
        ADD,    --00
        SUB,    --01
        OpAND,  --02
        OpOR,   --03
        OpXOR,  --04
        OpNAND, --05
        OpNOR,  --06
        OpXNOR, --07
        OpNOT,  --08
        OpSLL,  --09
        OpSRL,  --10
        others => (others => '0'));
begin
    salidaD <= memoria(conv_integer(funCode));
end Behavioral;
```

- **MopCode**

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

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity MopCode is
     generic (
        D : integer := 20;
        A : integer := 5);
    Port ( opCode : in  STD_LOGIC_VECTOR (A-1 downto 0);
           salidaD : out  STD_LOGIC_VECTOR (D-1 downto 0));
end MopCode;

architecture Behavioral of MOpcode is
--OPCODES--
--Tipo I
constant LI: std_logic_vector := x"00400";
constant LWI: std_logic_vector := x"04408";
constant LW: std_logic_vector := x"06531";
constant SWI: std_logic_vector := x"0800c";
constant SW: std_logic_vector := x"0A135";
constant ADDI: std_logic_vector := x"04533";
constant SUBI: std_logic_vector := x"04573";
constant ANDI: std_logic_vector := x"04503";
constant ORI: std_logic_vector := x"04513";
constant XORI: std_logic_vector := x"04523";
constant NANDI: std_logic_vector := x"045d3";
constant NORI: std_logic_vector := x"045c3";
constant XNORI: std_logic_vector := x"045a3";
constant BEQI: std_logic_vector := x"08071";
constant BNEI: std_logic_vector := x"08071";
constant BLTI: std_logic_vector := x"08071";
constant BLETI: std_logic_vector := x"08071";
constant BGTI: std_logic_vector := x"08071";
constant BGETI: std_logic_vector := x"08071";
constant SALTO: std_logic_vector := x"98333";--Saltos condicionales de
los B's tipo I

--Tipo J
constant B: std_logic_vector := x"10000";
constant CALL: std_logic_vector := x"50000";

--Otras Instrucciones
constant RET: std_logic_vector := x"20000";
constant NOP: std_logic_vector := x"00000";


type banco is array (0 to (2**A)-1) of std_logic_vector(D-1 downto 0);
constant memoria: banco := (
    SALTO,  --00
    LI,     --01
    LWI,    --02
    SWI,    --03
    SW,     --04
    ADDI,   --05
    SUBI,   --06
    ANDI,   --07
    ORI,    --08
    XORI,   --09
    NANDI,  --10
```

```
    NORI,    --11
    XNORI,   --12
    BEQI,    --13
    BNEI,    --14
    BLTI,    --15
    BLETI,   --16
    BGTI,    --17
    BGETI,   --18
    B,       --19
    CALL,    --20
    RET,     --21
    NOP,     --22
    LW,      --23
    others =>(others => '0'));
begin
    salidaD <= memoria(conv_integer(opCode));
end Behavioral;
```

- **Mux_SM**

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

entity Mux_SM is
    Port ( MfunCode, MopCode : in  STD_LOGIC_VECTOR (19 downto 0);
           SM : in  STD_LOGIC;
           Microinstruccion : out  STD_LOGIC_VECTOR (19 downto 0));
end Mux_SM;

architecture Behavioral of Mux_SM is
begin
    Microinstruccion <= MopCode when SM = '1' else MfunCode;
end Behavioral;
```

- **Mux_SODPC**

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

entity Mux_SODPC is
    Port ( opCode : in  STD_LOGIC_VECTOR (4 downto 0);
           SDOPC : in  STD_LOGIC;
           salida : out  STD_LOGIC_VECTOR (4 downto 0));
end Mux_SODPC;

architecture Behavioral of Mux_SODPC is
begin
    salida <= opCode when SDOPC = '1' else "00000";
end Behavioral;
```

- **Decodificador_Instruccion**

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

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

architecture Behavioral of Decodificador_Instruccion is
begin
    TIPOR <= '1' when opCode = "00000" else '0';
    BEQI <= '1' when opCode = "01101" else '0';
    BNEQI <= '1' when opCode = "01110" else '0';
    BLTI <= '1' when opCode = "01111" else '0';
    BLETI <= '1' when opCode = "10000" else '0';
    BGTI <= '1' when opCode = "10001" else '0';
    BGETI <= '1' when opCode = "10010" else '0';
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 A: std_logic := '0';
    signal B: std_logic := '0';
begin
    process (clk) begin
        if (clk'event and clk = '1') then --rising_edge(clk)
            A <= NOT A;
        end if;
        if (clk'event and clk = '0') then --falling_edge(clk)
            B <= NOT B;
        end if;
    end process;

    NA <= '0' when clr='1' else (A XOR B);
end Behavioral;
```

- **Regitro_Estado**

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

entity Registro_Estado is
    Port ( clk, clr, LF : in  STD_LOGIC;
           banderas : in  STD_LOGIC_VECTOR (3 downto 0);
           Q : out  STD_LOGIC_VECTOR (3 downto 0));
end Registro_Estado;
```

```
architecture Behavioral of Registro_Estado is
begin
    process(clk, clr) begin
        if clr = '1' then
            Q <= "0000";
        elsif (clk'event and clk='0') then
            if LF = '1' then
                Q <= banderas;
            end if;
        end if;
    end process;
end Behavioral;
```

- **Condicion**

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

entity Condicion is
    Port ( Q : in  STD_LOGIC_VECTOR (3 downto 0);
           EQ, NEQ, LT, LE, GTI, GET : out  STD_LOGIC);
end Condicion;

architecture Behavioral of Condicion is
begin
    --(0, N, Z, C)
    EQ <= '1' when Q= "0010" else '0';--Z
    NEQ <= not Q(1);--not(Z)
    LT <= Q(2);--not(C)
    LE <=(Q(2) or Q(1));--Z + not(C)
    GTI <= not Q(2);--not(Z) and C
    GET <=((not Q(2)) or Q(1));--C
end Behavioral;
```

- **Carta_ASM**

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

entity Carta_ASM is
    Port ( TIPOR, BEQ, BNEQ, BLT, BLE, BGT, BGET, NA : in  STD_LOGIC;
           EQ, NEQ, LT, LE, GTI, GET : in  STD_LOGIC;
           clk, clr : in  STD_LOGIC;
           SDOPC, SM : out  STD_LOGIC);
end Carta_ASM;

architecture Behavioral of Carta_ASM is
begin

estados: process (clk, clr, TIPOR, BEQ, BNEQ, BLT, BLE, BGT, BGET, NA,
EQ, NEQ, LT, LE, GTI, GET)
    begin
        if clr = '1' then
            SDOPC <= '0';
            SM <= '0';
```

```
elsif rising_edge(clk) then
    if TIPOR = '1' then
        SDOPC <= '0';
        SM <= '0';
    else
        if BEQ = '1' then --BEQ
            if NA = '1' then --VERIFICACION
                SDOPC <= '0';
                SM <= '1';
        elsif EQ = '1' then --SALTO
                SDOPC <= '1';
                SM <= '1';
        else
                SDOPC <= '0';
                SM <= '1';
        end if;
    elsif BNEQ = '1' then--BNEQ
        if NA = '1' then --VERIFICACION
            SDOPC <= '0';
            SM <= '1';
        elsif NEQ = '1' THEN --SALTO
            SDOPC <= '1';
            SM <= '1';
        else
            SDOPC <= '0';
            SM <= '1';
        end if;
    elsif BLT = '1' then --BLT
        if NA = '1' then --VERIFICACION
            SDOPC <= '0';
            SM <= '1';
        elsif LT = '1' then --SALTO
            SDOPC <= '1';
            SM <= '1';
        else
            SDOPC <= '0';
            SM <= '1';
        end if;
    elsif BLE = '1' then --BLE
        if NA = '1' then --VERIFICACION
            SDOPC <= '0';
            SM <= '1';
        elsif LE = '1' then --SALTO
            SDOPC <= '1';
            SM <= '1';
        else
            SDOPC <= '0';
            SM <= '1';
        end if;
    elsif BGT = '1' then --BGT
        if NA = '1' then --VERIFICACION
            SDOPC <= '0';
            SM <= '1';
        elsif GTI = '1' then --SALTO
            SDOPC <= '1';
            SM <= '1';
        else
```

```
                    SDOPC <= '0';
                    SM <= '1';
                end if;
            elsif BGET = '1' then --BGET
                if NA = '1' then --VERIFICACION
                    SDOPC <= '0';
                    SM <= '1';
                elsif GET = '1' then --SALTO
                    SDOPC <= '1';
                    SM <= '1';
                else
                    SDOPC <= '0';
                    SM <= '1';
                end if;
            end if;
        end if;
    end if;
end process;

end Behavioral;
```

Código de Simulación

- **MfunCode**

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity MfunCode_tb is
end MfunCode_tb;

architecture Behavioral of MfunCode_tb is

component MfunCode is
    Port ( funCode : in STD_LOGIC_VECTOR (3 downto 0);
           salidaD : out STD_LOGIC_VECTOR (19 downto 0));
end component;

--Inputs
signal funCode : STD_LOGIC_VECTOR(3 downto 0) := (others => '0');

--Outputs
signal salidaD : STD_LOGIC_VECTOR(19 downto 0);

begin
    -- Instantiate the Unit Under Test (UUT)
    uut: MfunCode
    Port Map ( funCode => funCode,
               salidaD => salidaD);

    -- Stimulus process
    stim_proc: process
    begin
        funCode <= x"0";
        wait for 50 ns;
        funCode <= x"1";
```

```
        wait for 50 ns;
        funCode <= x"2";
        wait for 50 ns;
        funCode <= x"3";
        wait for 50 ns;
        funCode <= x"4";
        wait for 50 ns;
        funCode <= x"5";
        wait for 50 ns;
        funCode <= x"6";
        wait for 50 ns;
        funCode <= x"7";
        wait for 50 ns;
        funCode <= x"8";
        wait for 50 ns;
        funCode <= x"9";
        wait for 50 ns;
        funCode <= x"A";
        wait for 50 ns;
        funCode <= x"B";
        wait for 50 ns;
        funCode <= x"C";
        wait for 50 ns;
        funCode <= x"D";
        wait for 50 ns;
        funCode <= x"E";
        wait for 50 ns;
        funCode <= x"F";
        wait for 50 ns;
        wait;
    end process;
end Behavioral;
```

- **MopCode**

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity MopCode_tb is
end MopCode_tb;

architecture Behavioral of MopCode_tb is

component MopCode is
    Port ( opCode : in  STD_LOGIC_VECTOR (4 downto 0);
           salidaD : out  STD_LOGIC_VECTOR (19 downto 0));
end component;

--Inputs
signal opCode : STD_LOGIC_VECTOR(4 downto 0) := (others => '0');

--Outputs
signal salidaD : STD_LOGIC_VECTOR(19 downto 0);

begin
    -- Instantiate the Unit Under Test (UUT)
```

```
    uut: MopCode
        Port Map ( opCode => opCode,
                   salidaD => salidaD);

    -- Stimulus process
    stim_proc: process
    begin
        opCode <= "00001";--LI
        wait for 50 ns;
        opCode <= "00011";--SWI
        wait for 50 ns;
        opCode <= "00101";--ADDI
        wait for 50 ns;
        opCode <= "00110";--SUBI
        wait for 50 ns;
        opCode <= "01001";--XORI
        wait for 50 ns;
        opCode <= "01100";--XNORI
        wait for 50 ns;
        opCode <= "01111";--BLTI
        wait for 50 ns;
        opCode <= "10010";--BGETI
        wait for 50 ns;
        opCode <= "10011";--B
        wait for 50 ns;
        opCode <= "10100";--CALL
        wait for 50 ns;
        opCode <= "10101";--RET
        wait for 50 ns;
        opCode <= "10110";--NOP
        wait for 50 ns;
        wait;
    end process;
end Behavioral;
```

- **Decodificador_Instruccion**

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

entity Decodificador_Instruccion_tb is
end Decodificador_Instruccion_tb;

architecture Behavioral of Decodificador_Instruccion_tb is

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

--Inputs
signal opCode :STD_LOGIC_VECTOR(4 downto 0) := (others => '0');

--Outputs
signal TIPOR : STD_LOGIC;
signal BEQI : STD_LOGIC;
```

```
signal BNEQI : STD_LOGIC;
signal BLTI : STD_LOGIC;
signal BLETI : STD_LOGIC;
signal BGTI : STD_LOGIC;
signal BGETI : STD_LOGIC;

begin
    uut: Decodificador_Instruccion
    Port Map ( opCode => opCode,
               TIPOR => TIPOR,
               BEQI => BEQI,
               BNEQI => BNEQI,
               BLTI => BLTI,
               BLETI => BLETI,
               BGTI => BGTI,
               BGETI => BGETI);

    -- Stimulus process
    stim_proc: process
    begin
        opCode <= "00001";--LI
        wait for 50 ns;
        opCode <= "00000";--Tipo R
        wait for 50 ns;
        opCode <= "01101";--BEQI
        wait for 50 ns;
        opCode <= "01110";--BNEI
        wait for 50 ns;
        opCode <= "01111";--BLTI
        wait for 50 ns;
        opCode <= "10000";--BLETI
        wait for 50 ns;
        opCode <= "10001";--BGTI
        wait for 50 ns;
        opCode <= "10010";--BGETI
        wait for 50 ns;
        wait;
   end process;
end Behavioral;
```

- **Nivel**

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

entity Nivel_tb is
end Nivel_tb;

architecture Behavioral of Nivel_tb is

component Nivel is
    Port ( clk, clr : in  STD_LOGIC;
           NA : out  STD_LOGIC);
end component;

--Inputs
```

```
signal clk : STD_LOGIC := '0';
signal clr : STD_LOGIC := '0';

--Outputs
signal NA : STD_LOGIC;

-- Clock period definitions
constant clk_period : time := 10 ns;

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: Nivel
    Port Map ( clk => clk,
               clr => clr,
               NA => NA);

    -- Clock process definitions
    clk_process :process
    begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;

    -- Stimulus process
    stim_proc: process
    begin
        wait for 20 ns;
        clr <= '1';
        wait for 20 ns;
        clr <= '0';
        wait for 40 ns;
        wait for 20 ns;
        clr <= '1';
        wait for 25 ns;
        clr <= '0';
        wait for 40 ns;
        wait;
   end process;

end Behavioral;
```

- **Regitro_Estado**

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

entity Registro_Estado_tb is
end Registro_Estado_tb;

architecture Behavioral of Registro_Estado_tb is

component Registro_Estado is
    Port ( clk, clr, LF : in  STD_LOGIC;
```

```
            banderas : in  STD_LOGIC_VECTOR (3 downto 0);
            Q : out  STD_LOGIC_VECTOR (3 downto 0));
end component;

--Inputs
signal clk : STD_LOGIC := '0';
signal clr : STD_LOGIC := '0';
signal LF : STD_LOGIC := '0';
signal banderas : STD_LOGIC_VECTOR(3 downto 0) := (others => '0');

--Outputs
signal Q : STD_LOGIC_VECTOR(3 downto 0);

-- Clock period definitions
constant CLK_period : time := 10 ns;

begin
    -- Instantiate the Unit Under Test (UUT)
    uut: Registro_Estado
        Port Map ( clk => clk,
                   clr => clr,
                   LF => LF,
                   banderas => banderas,
                   Q => Q);
    -- Clock process definitions
    CLK_process :process
    begin
        clk <= '0';
        wait for CLK_period/2;
        clk <= '1';
        wait for CLK_period/2;
    end process;

    -- Stimulus process
    stim_proc: process
    begin
        clr <= '1';
        wait for 20 ns;
        clr <= '0';
        banderas <= "0110";
        LF <= '1';
        wait for 50 ns;
        banderas <= "1010";
        LF <= '0';
        wait for 50 ns;
        LF <= '1';
        wait for 50 ns;
        banderas <= "1001";
        LF <= '0';
        wait for 50 ns;
        LF <= '1';
        wait for 50 ns;
        wait;
    end process;
end Behavioral;
```

- **Condicion**

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

entity Condicion_tb IS
end Condicion_tb;

architecture Behavioral of Condicion_tb is

component Condicion is
    Port ( Q : in  STD_LOGIC_VECTOR (3 downto 0);
           EQ, NEQ, LT, LE, GTI, GET : out  STD_LOGIC);
end component;

--Inputs
signal Q : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');
--Outputs
signal EQ : STD_LOGIC;
signal NEQ : STD_LOGIC;
signal LT : STD_LOGIC;
signal LE : STD_LOGIC;
signal GTI : STD_LOGIC;
signal GET : STD_LOGIC;

begin
    -- Instantiate the Unit Under Test (UUT)
    uut: Condicion
        Port Map ( Q => Q,
                   EQ => EQ,
                   NEQ => NEQ,
                   LT => LT,
                   LE => LE,
                   GTI => GTI,
                   GET => GET);

   -- Stimulus process
   stim_proc: process
   begin
      Q <= "0000";
      wait for 50 ns;
      Q <= "0010";
      wait for 50 ns;
      Q <= "0100";
      wait for 50 ns;
      Q <= "1000";
      wait for 50 ns;
      Q <= "0110";
      wait for 50 ns;
      Q <= "1001";
      wait for 50 ns;
      Q <= "1010";
      wait for 50 ns;
      wait;
   end process;
end Behavioral;
```

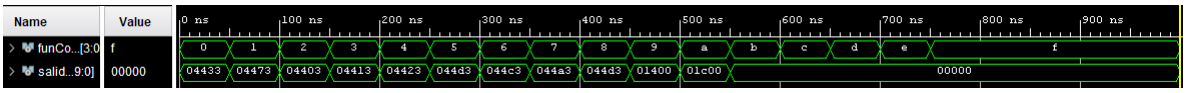
- **Carta_ASM**

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

entity Carta_ASM_tb is
end Carta_ASM_tb;

architecture Behavioral of Carta_ASM_tb is

component Carta_ASM is
    Port ( TIPOR, BEQ, BNEQ, BLT, BLE, BGT, BGET, NA : in STD_LOGIC;
           EQ, NEQ, LT, LE, GTI, GET : in STD_LOGIC;
           clk, clr : in STD_LOGIC;
           SDOPC, SM : out STD_LOGIC);
end component;

--Inputs
signal TIPOR : STD_LOGIC := '0';
signal BEQ : STD_LOGIC := '0';
signal BNEQ : STD_LOGIC := '0';
signal BLT : STD_LOGIC := '0';
signal BLE : STD_LOGIC := '0';
signal BGT : STD_LOGIC := '0';
signal BGET : STD_LOGIC := '0';
signal NA : STD_LOGIC := '0';
signal EQ : STD_LOGIC := '0';
signal NEQ : STD_LOGIC := '0';
signal LT : STD_LOGIC := '0';
signal LE : STD_LOGIC := '0';
signal GTI : STD_LOGIC := '0';
signal GET : STD_LOGIC := '0';
signal clk : STD_LOGIC := '0';
signal clr : STD_LOGIC := '0';

--Outputs
signal SDOPC : STD_LOGIC;
signal SM : STD_LOGIC;

-- Clock period definitions
constant CLK_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: Carta_ASM
        Port Map ( TIPOR => TIPOR,
                   BEQ => BEQ,
                   BNEQ => BNEQ,
                   BLT => BLT,
                   BLE => BLE,
                   BGT => BGT,
                   BGET => BGET,
                   NA => NA,
                   EQ => EQ,
                   NEQ => NEQ,
                   LT => LT,
```

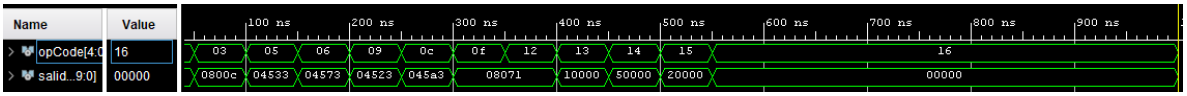
```
                    LE => LE,
                    GTI => GTI,
                    GET => GET,
                    clk => clk,
                    clr => clr,
                    SDOPC => SDOPC,
                    SM => SM);

    -- Clock process definitions
    CLK_process :process
    begin
        clk <= '0';
        wait for CLK_period/2;
        clk <= '1';
        wait for CLK_period/2;
    end process;

    stim_proc: process
    begin
        clr <= '1';
        wait for 20 ns;
        clr <= '0';
        BEQ <= '1';
        EQ <= '1';
        wait for CLK_period*5;
        wait;
    end process;
end Behavioral;
```

Simulación:

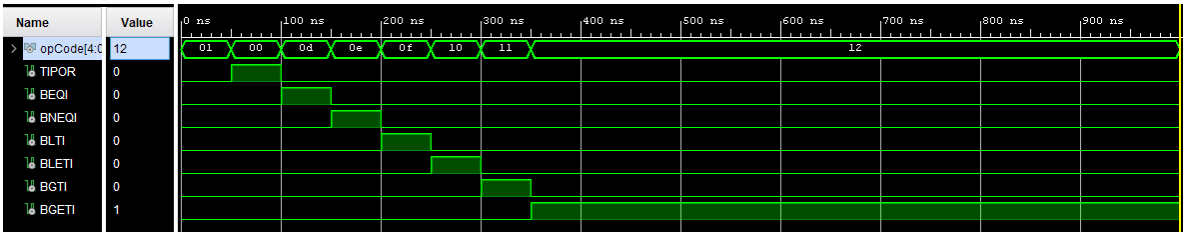
- **MfunCode**



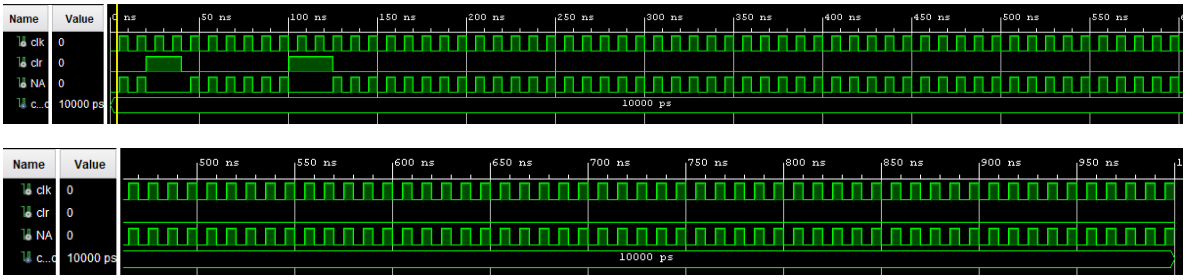
- **MopCode**



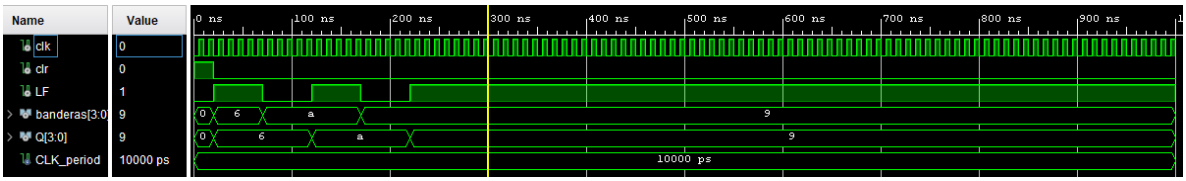
- **Decodificador_Instruccion**



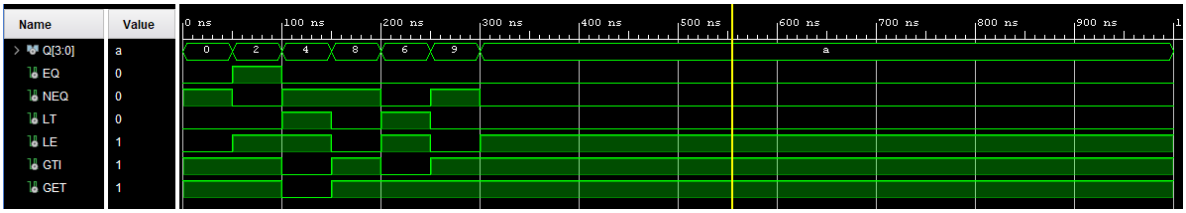
- **Nivel**



- **Regitro_Estado**



- **Condición**



- **Carta_ASM**

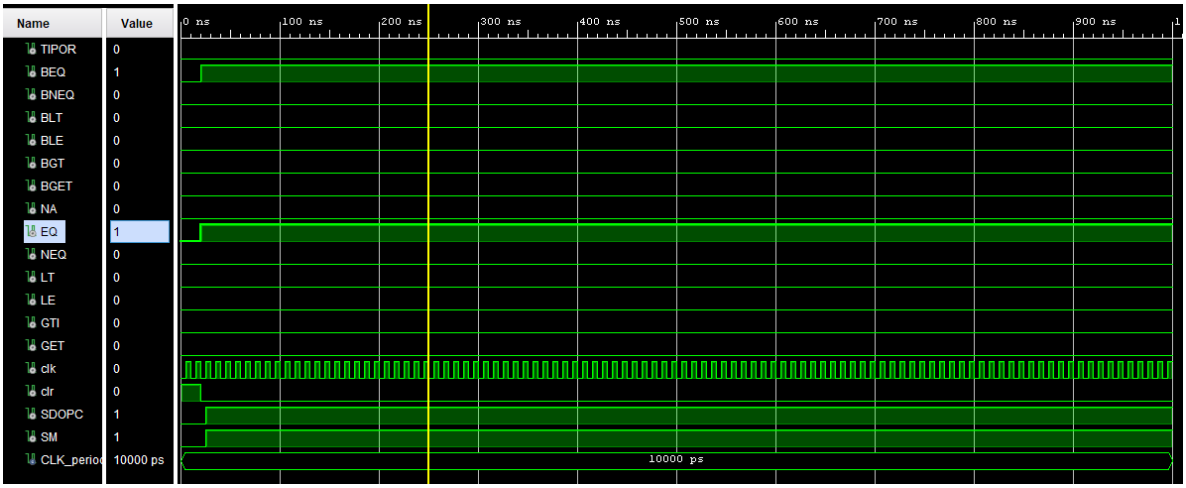
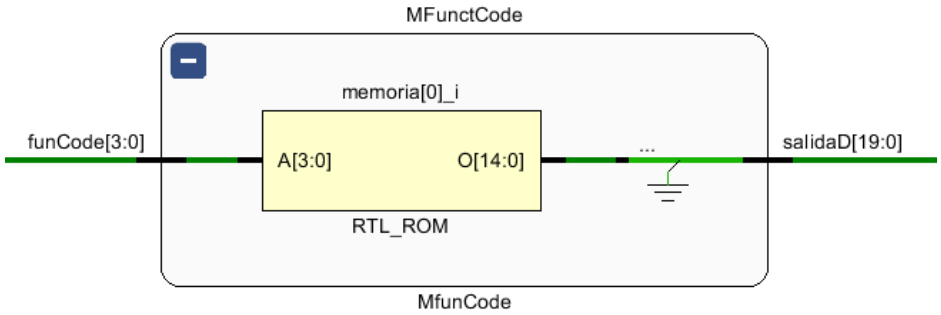
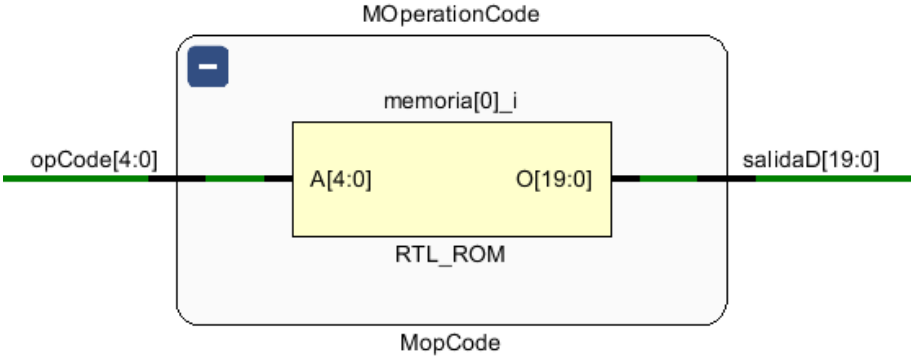


Diagrama RTL:

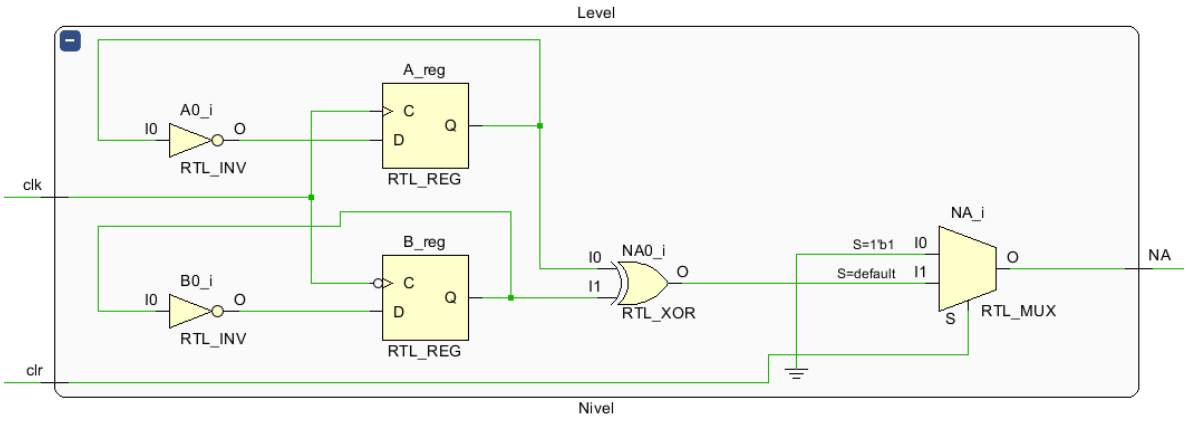
- **MfunCode**



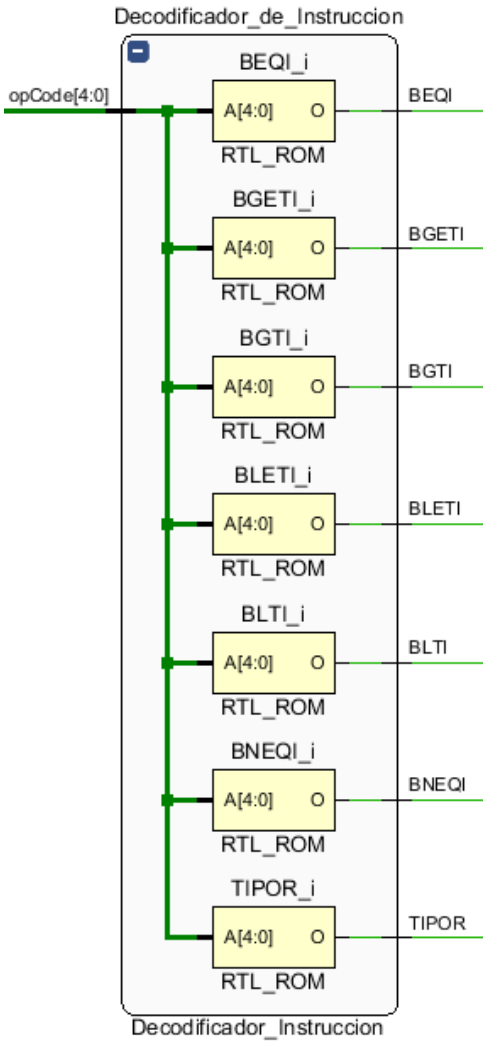
- **MopCode**



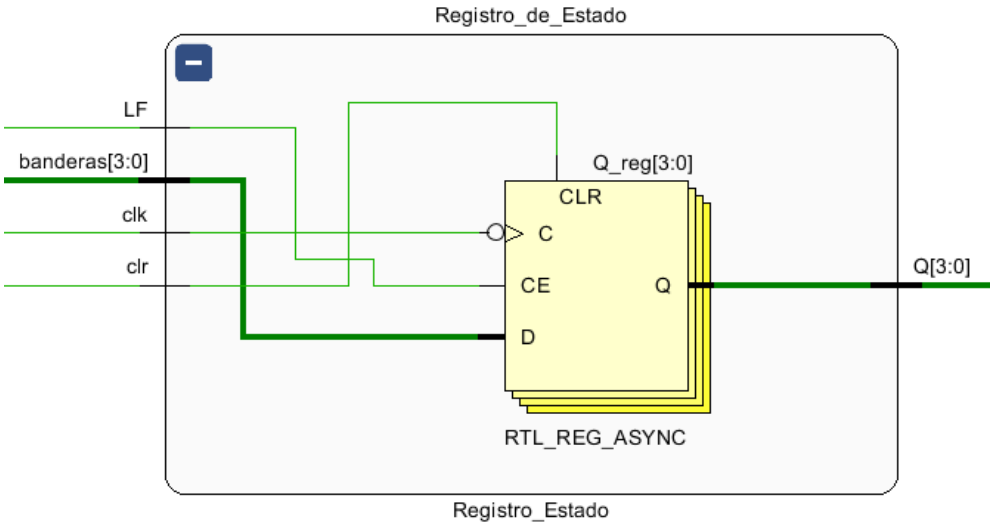
- **Nivel**



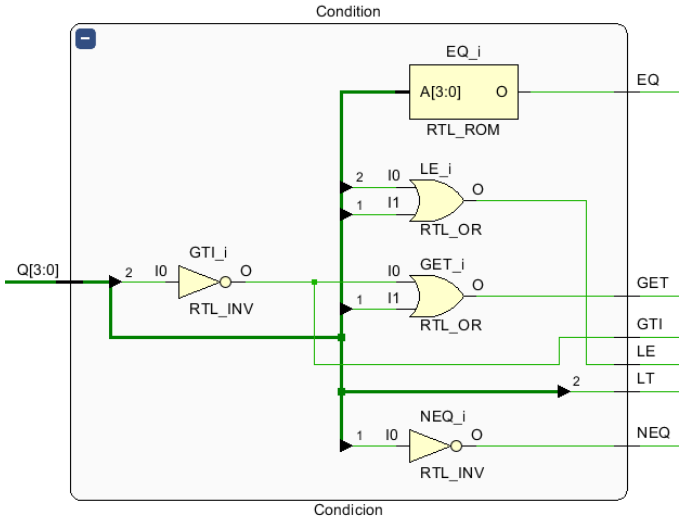
- **Decodificador_Instruccion**



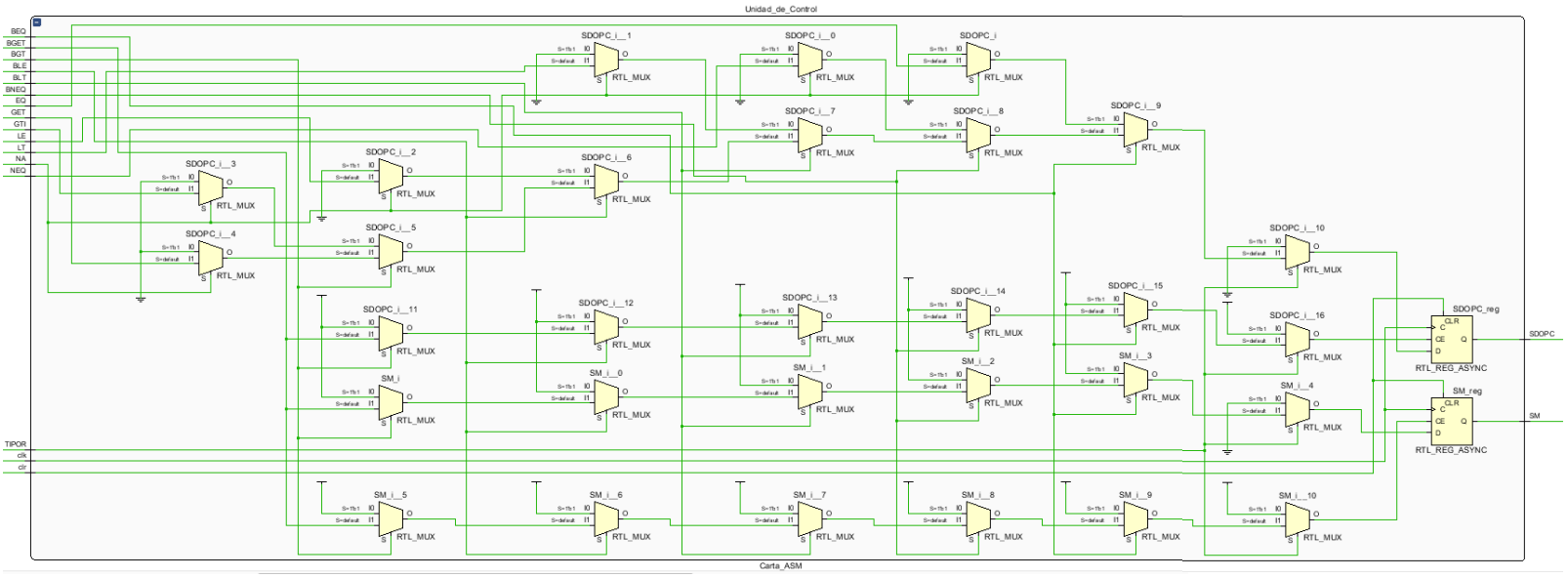
- **Regitro_Estado**



- **Condicion**



- **Carta_ASM**



Arquitectura Completa

Código de Implementación

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

entity Unidad_Control is
    Port ( funCode, banderas : in  STD_LOGIC_VECTOR (3 downto 0);
           opCode : in  STD_LOGIC_VECTOR (4 downto 0);
           clk, clr, LF: in  STD_LOGIC;
           Microinstruccion : out  STD_LOGIC_VECTOR (19 downto 0);
           LVL: OUT STD_LOGIC);
end Unidad_Control;

architecture Behavioral of Unidad_Control is

component Carta_ASM is
    Port ( TIPOR, BEQ, BNEQ, BLT, BLE, BGT, BGET, NA : in  STD_LOGIC;
           EQ, NEQ, LT, LE, GTI, GET : in  STD_LOGIC;
           clk, clr : in  STD_LOGIC;
           SDOPC, SM : out  STD_LOGIC);
end component;

component MfunCode is
    Port ( funCode : in  STD_LOGIC_VECTOR (3 downto 0);
           salidaD : out  STD_LOGIC_VECTOR (19 downto 0));
end component;

component MopCode is
    Port ( opCode : in  STD_LOGIC_VECTOR (4 downto 0);
           salidaD : out  STD_LOGIC_VECTOR (19 downto 0));
end component;

component Mux_SODPC is
    Port ( opCode : in  STD_LOGIC_VECTOR (4 downto 0);
           SDOPC : in  STD_LOGIC;
           salida : out  STD_LOGIC_VECTOR (4 downto 0));
end component;

component Mux_SM is
    Port ( MfunCode, MopCode : in  STD_LOGIC_VECTOR (19 downto 0);
           SM : in  STD_LOGIC;
           Microinstruccion : out  STD_LOGIC_VECTOR (19 downto 0));
end component;

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

component Nivel is
    Port ( clk, clr : in  STD_LOGIC;
           NA : out  STD_LOGIC);
end component;
```

```
component Registro_Estado is
    Port ( clk, clr, LF : in  STD_LOGIC;
           banderas : in  STD_LOGIC_VECTOR (3 downto 0);
           Q : out  STD_LOGIC_VECTOR (3 downto 0));
end component;

component Condicion is
    Port ( Q : in  STD_LOGIC_VECTOR (3 downto 0);
           EQ, NEQ, LT, LE, GTI, GET : out  STD_LOGIC);
end component;

signal OpR, OpNoR: STD_LOGIC_VECTOR(19 DOWNTO 0);
signal SM_Mux: STD_LOGIC_VECTOR(4 DOWNTO 0);
signal Q: STD_LOGIC_VECTOR(3 DOWNTO 0);
signal SDOPC, NA, SM: STD_LOGIC;
signal TIPOR, BEQI, BNEQI, BLTI, BLETI,BGTI,BGETI: STD_LOGIC;
signal EQ, NEQ, LT, LE, GTI, GET : STD_LOGIC;
begin

MFunctCode: MFunCode
    Port map ( funCode => funCode,
               salidaD => OpR);

MOperationCode: MOpCode
    Port map ( opCode => SM_Mux,
               salidaD => OpNoR);

MuxSM: Mux_SM
    Port map ( MfunCode => OpR,
               MopCode => OpNoR,
               SM => SM,
               Microinstruccion => Microinstruccion);

MuxSODPC: Mux_SODPC
    Port map ( opCode => opCode,
               SDOPC => SDOPC,
               salida => SM_Mux);

Decodificador_de_Instruccion: Decodificador_Instruccion
    Port map( opCode => opCode,
              TIPOR => TIPOR,
              BEQI => BEQI,
              BNEQI => BNEQI,
              BLTI => BLTI,
              BLETI => BLETI,
              BGTI => BGTI,
              BGETI => BGETI);

Level: Nivel
    Port map ( clk => clk,
               clr => clr,
               NA => NA);

Registro_de_Estado: Registro_Estado
    Port map( clk => clk,
              clr => clr,
              LF => LF,
```

```
                banderas => banderas,
                Q => Q);

Condition: Condicion
    Port map ( Q => Q,
               EQ => EQ,
               NEQ => NEQ,
               LT => LT,
               LE => LE,
               GTI => GTI,
               GET => GET);

Unidad_de_Control: Carta_ASM
    Port map ( TIPOR => TIPOR,
               BEQ => BEQI,
               BNEQ => BNEQI,
               BLT => BLTI,
               BLE => BLETI,
               BGT => BGTI,
               BGET => BGETI,
               NA => NA,
               EQ => EQ,
               NEQ => NEQ,
               LT => LT,
               LE => LE,
               GTI => GTI,
               GET => GET,
               clk => clk,
               clr => clr,
               SDOPC => SDOPC,
               SM => SM);
LVL <= NA;
end Behavioral;
```

Código de Simulación

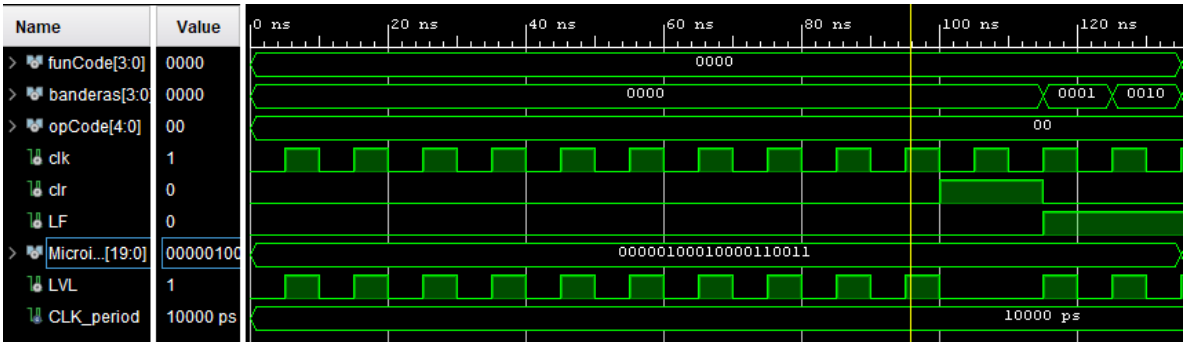
```
library IEEE;
LIBRARY STD;
use STD.TEXTIO.ALL;
use IEEE.STD_LOGIC_TEXTIO.ALL;  --PERMITE USAR STD_LOGIC
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity Unidad_Control_tb is
end Unidad_Control_tb;

architecture Behavioral of Unidad_Control_tb is

component Unidad_Control is
    Port ( funCode, banderas : in  STD_LOGIC_VECTOR (3 downto 0);
           opCode : in  STD_LOGIC_VECTOR (4 downto 0);
           clk, clr, LF: in  STD_LOGIC;
           Microinstruccion : out  STD_LOGIC_VECTOR (19 downto 0);
           LVL: OUT STD_LOGIC);
end component;
```

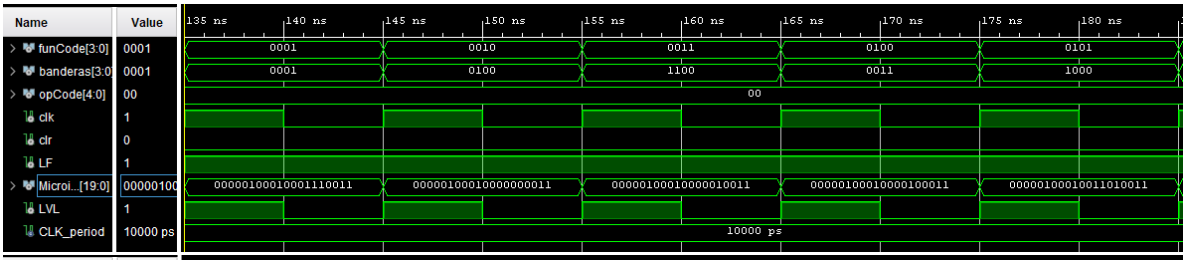
```
--Inputs
signal funCode : std_logic_vector(3 downto 0) := (others => '0');
signal banderas : std_logic_vector(3 downto 0) := (others => '0');
signal opCode : std_logic_vector(4 downto 0) := (others => '0');
signal clk : std_logic := '0';
signal clr : std_logic := '0';
signal LF : std_logic := '0';

--Outputs
signal Microinstruccion : std_logic_vector(19 downto 0);
signal LVL: STD_LOGIC;

-- Clock period definitions
constant CLK_period : time := 10 ns;

begin
    -- Instantiate the Unit Under Test (UUT)
    uut: Unidad_Control
        Port Map ( funCode => funCode,
                   banderas => banderas,
                   opCode => opCode,
                   clk => clk,
                   clr => clr,
                   LF => LF,
                   Microinstruccion => Microinstruccion,
                   LVL => LVL);

    -- Clock process definitions
    CLK_process :process
    begin
        CLK <= '0';
        wait for CLK_period/2;
        CLK <= '1';
        wait for CLK_period/2;
    end process;

    -- Stimulus process
    stim_proc: process
        file ARCH_RES : TEXT;--Archivo de resultados
        variable LINEA_RES : line;--Linea de resultado
        file ARCH_VEC : TEXT;--Archivo de vectores
        variable LINEA_VEC : line;--Linea de vectores

        --Variables
        variable V_funCode, V_banderas: STD_LOGIC_VECTOR(3 DOWNTO 0);
        variable V_opCode: STD_LOGIC_VECTOR(4 DOWNTO 0);
        variable V_clr, V_LF, V_LVL: STD_LOGIC;
        variable V_Microinstruccion: std_logic_vector(19 downto 0);
        --Cadena
        VARIABLE CADENA : STRING(1 TO 7);

        begin
            file_open(ARCH_VEC, "VECTORES.txt", READ_MODE);
            file_open(ARCH_RES, "RESULTADO.txt", WRITE_MODE);
            --Impresion de Cadenas
            CADENA := "OP_CODE";
```

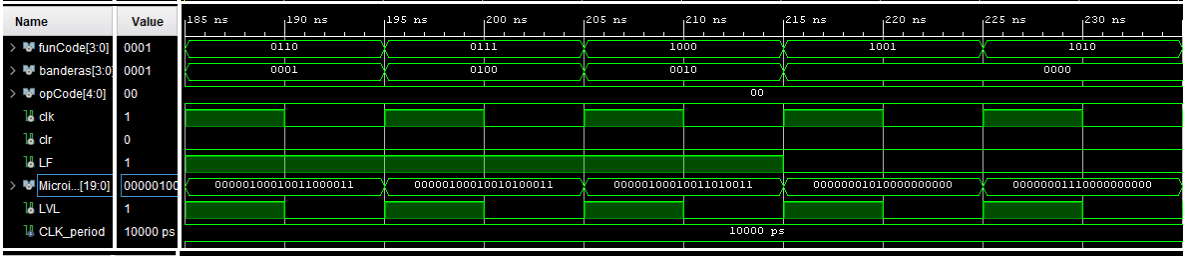
```
            write(LINEA_RES, CADENA, right, CADENA'LENGTH+1); --OP_CODE
            CADENA := "FUNCODE";
            write(LINEA_RES, CADENA, right, CADENA'LENGTH+1); --FUNC_CODE
            CADENA := "  FLAGS";
            write(LINEA_RES, CADENA, right, CADENA'LENGTH+1); --BANDERAS
            CADENA := "    CLR";
            write(LINEA_RES, CADENA, right, CADENA'LENGTH+1); --CLR
            CADENA := "     LF";
            write(LINEA_RES, CADENA, right, CADENA'LENGTH+1); --LF
            CADENA := "MICROIN";
            write(LINEA_RES, CADENA, right, CADENA'LENGTH+14); --
MICROINSTRUCCION
            CADENA := "  NIVEL";
            write(LINEA_RES, CADENA, right, CADENA'LENGTH+1); --NIVEL
            writeline(ARCH_RES,LINEA_RES);--Escribe la linea en el
archivo

            --Impresion de Resultados
            wait for 100 ns;

            for j in 0 to 46 loop
                --Lectura de VECTORES.TXT
                readline(ARCH_VEC,LINEA_VEC); -- lee una linea completa
                read(LINEA_VEC, V_opCode);
                opCode <= V_opCode;
                read(LINEA_VEC, V_funCode);
                funCode <= V_funCode;
                read(LINEA_VEC, V_banderas);
                banderas <= V_banderas;
                read(LINEA_VEC, v_clr);
                clr <= V_clr;
                read(LINEA_VEC, V_LF);
                LF <= V_LF;

                wait until RISING_EDGE(CLK);--Flanco de subida
                V_Microinstruccion := Microinstruccion; -- Asignacion de
Salida
                V_LVL := LVL;
                --Escritura de Resultados
                write(LINEA_RES, V_opCode, right, 8);
                write(LINEA_RES, V_funCode, right, 8);
                write(LINEA_RES, V_banderas, right, 8);
                write(LINEA_RES, V_CLR, right, 8);
                write(LINEA_RES, V_LF, right, 8);
                write(LINEA_RES, V_Microinstruccion, right, 21);

                if(V_LVL = '1') then
                    CADENA := "   ALTO";
                    write(LINEA_RES, CADENA, right, 8);
                else
                    CADENA := "   BAJO";
                    write(LINEA_RES, CADENA, right, 8);
                end if;

                writeline(ARCH_RES,LINEA_RES);--Escribe la linea en el
archivo
            end loop;
```

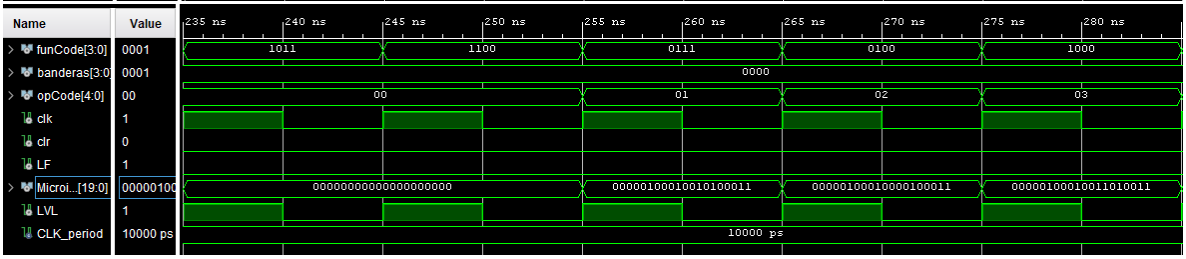
```
            file_close(ARCH_VEC);--Cierra el archivo
            file_close(ARCH_RES);--Cierra el archivo
            wait;
    end process;-- Stimulus process
end Behavioral;
```

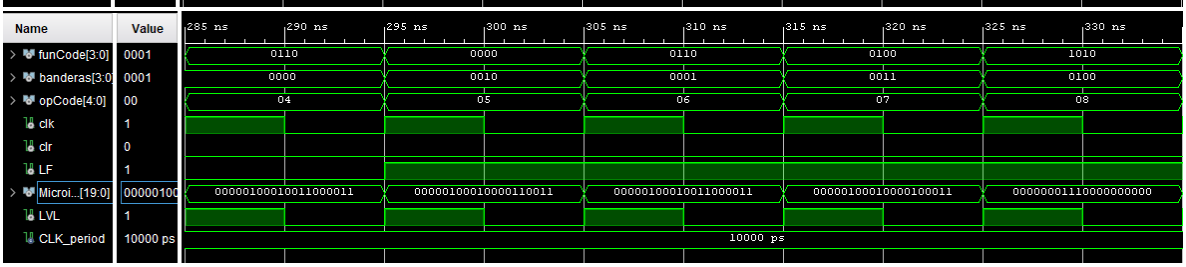
Simulación:











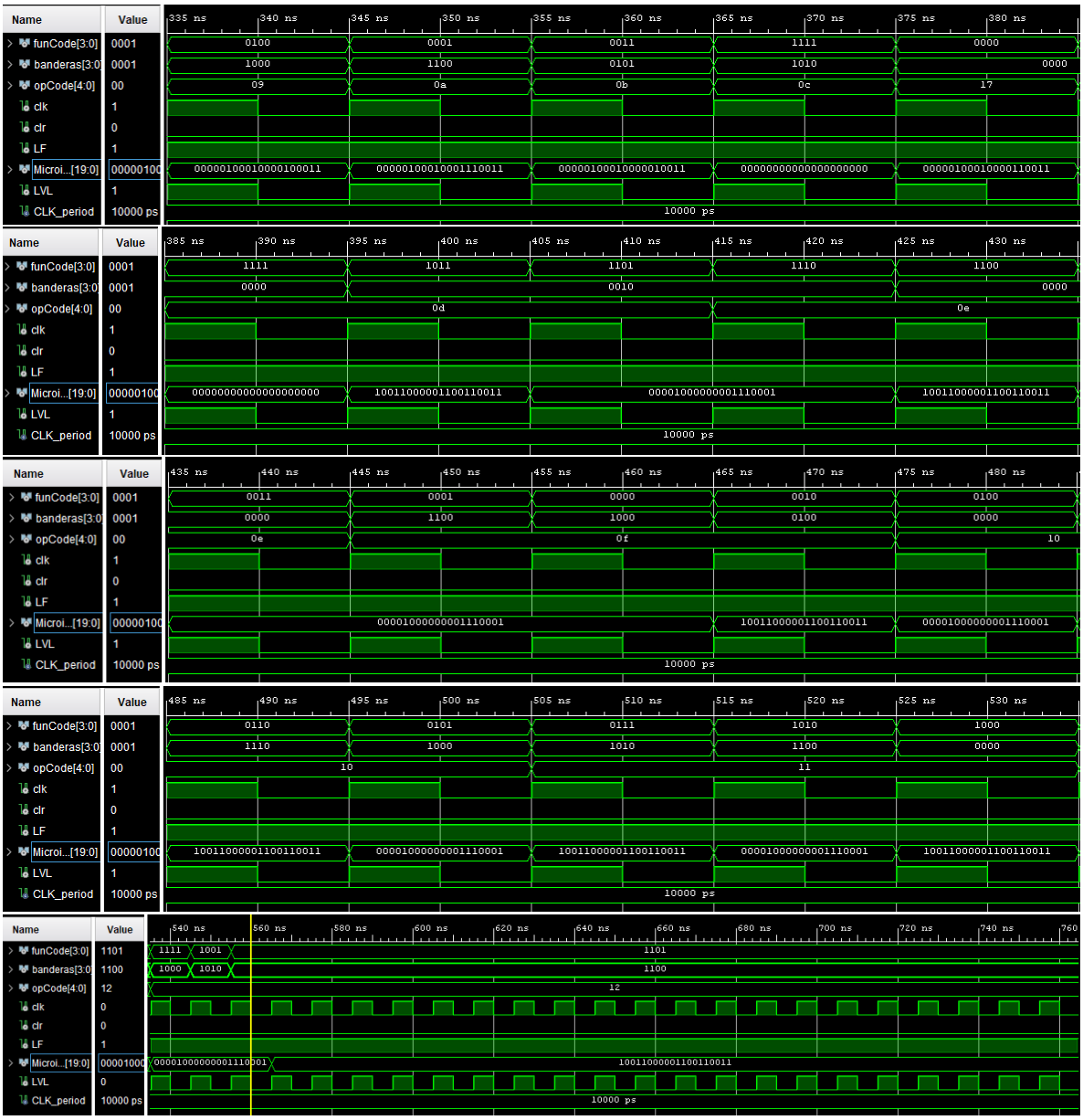
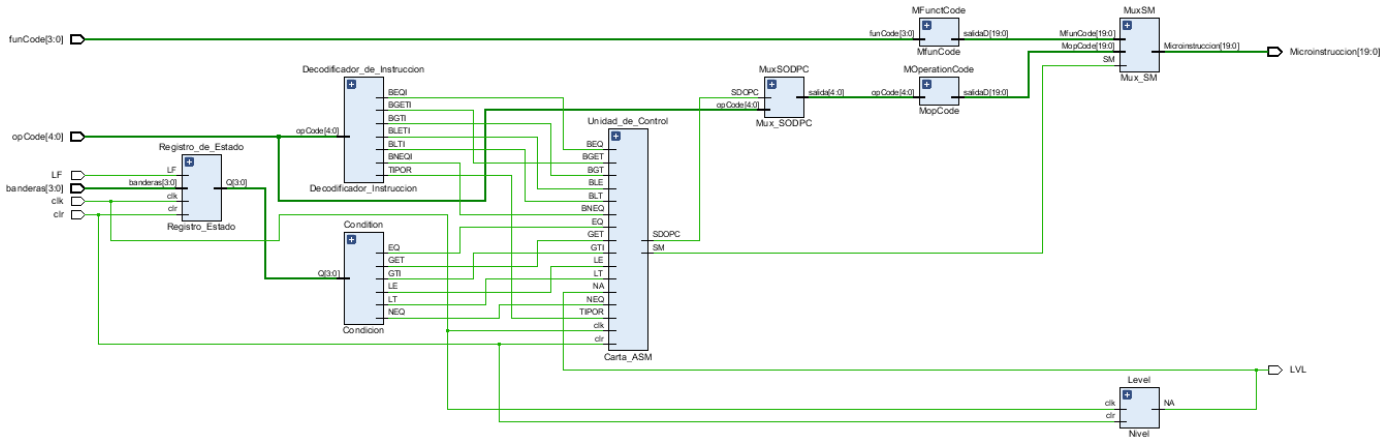


Diagrama RTL:



Archivo de Entradas:


```
00000 0000 0000 1 0
00000 0000 0000 1 0
00000 0000 0001 0 1
00000 0000 0010 0 1
00000 0001 0001 0 1
00000 0010 0100 0 1
00000 0011 1100 0 1
00000 0100 0011 0 1
00000 0101 1000 0 1
00000 0110 0001 0 1
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
```

Archivo de Salidas:

RESULTADO: Bloc de notas

Archivo Edición Formato Ver Ayuda

OP_CODE	FUNCODE	FLAGS	CLR	LF	MICROIN	NIVEL
00000	0000	0000	1	0	00000100010000110011	BAJO
00000	0000	0000	1	0	00000100010000110011	BAJO
00000	0000	0001	0	1	00000100010000110011	BAJO
00000	0000	0010	0	1	00000100010000110011	BAJO
00000	0001	0001	0	1	00000100010001110011	BAJO
00000	0010	0100	0	1	00000100010000000011	BAJO
00000	0011	1100	0	1	00000100010000010011	BAJO
00000	0100	0011	0	1	00000100010000100011	BAJO
00000	0101	1000	0	1	00000100010011010011	BAJO
00000	0110	0001	0	1	00000100010011000011	BAJO
00000	0111	0100	0	1	00000100010010100011	BAJO
00000	1000	0010	0	1	00000100010011010011	BAJO
00000	1001	0000	0	0	00000000101000000000	BAJO
00000	1010	0000	0	0	00000000111000000000	BAJO
00000	1011	0000	0	0	00000000000000000000	BAJO
00000	1100	0000	0	0	00000000000000000000	BAJO
00001	0111	0000	0	0	00000100010010100011	BAJO
00010	0100	0000	0	0	00000100010000100011	BAJO
00011	1000	0000	0	0	00000100010011010011	BAJO
00100	0110	0000	0	0	00000100010011000011	BAJO
00101	0000	0010	0	1	00000100010000110011	BAJO
00110	0110	0001	0	1	00000100010011000011	BAJO
00111	0100	0011	0	1	00000100010000100011	BAJO
01000	1010	0100	0	1	00000000111000000000	BAJO
01001	0100	1000	0	1	00000100010000100011	BAJO
01010	0001	1100	0	1	00000100010001110011	BAJO
01011	0011	0101	0	1	00000100010000010011	BAJO
01100	1111	1010	0	1	00000000000000000000	BAJO
10111	0000	0000	0	1	00000100010000110011	BAJO
01101	1111	0000	0	1	00000000000000000000	BAJO
01101	1011	0010	0	1	10011000001100110011	BAJO
01101	1101	0010	0	1	00001000000001110001	BAJO
01110	1110	0010	0	1	00001000000001110001	BAJO
01110	1100	0000	0	1	10011000001100110011	BAJO
01110	0011	0000	0	1	00001000000001110001	BAJO
01111	0001	1100	0	1	00001000000001110001	BAJO
01111	0000	1000	0	1	00001000000001110001	BAJO
01111	0010	0100	0	1	10011000001100110011	BAJO
10000	0100	0000	0	1	00001000000001110001	BAJO
10000	0110	1110	0	1	10011000001100110011	BAJO
10000	0101	1000	0	1	00001000000001110001	BAJO
10001	0111	1010	0	1	10011000001100110011	BAJO
10001	1010	1100	0	1	00001000000001110001	BAJO
10001	1000	0000	0	1	10011000001100110011	BAJO
10010	1111	1000	0	1	00001000000001110001	BAJO
10010	1001	1010	0	1	00001000000001110001	BAJO
10010	1101	1100	0	1	00001000000001110001	BAJO