Reporte de practica 8

González Pardo Adrian

Abril 2020

1. Código VHDL

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

entity memPrograma is
    generic(
        m:integer:= 10;--tamanio del bus de direcciones
        n:integer:= 25--tamanio de palabra
    );
    Port(dir:in STD_LOGIC_VECTOR (m-1 downto 0);
         inst:out STD_LOGIC_VECTOR (n-1 downto 0));
end memPrograma;

architecture Behavioral of memPrograma is
        --C-OPERACION
    constant TYPER: std_logic_vector(4 downto 0):="00000";
    constant LI: std_logic_vector(4 downto 0):="00001";
    constant LWI: std_logic_vector(4 downto 0):="00010";
    constant LW: std_logic_vector(4 downto 0):="10111";
    constant SWI: std_logic_vector(4 downto 0):="00011";
    constant SW: std_logic_vector(4 downto 0):="00100";
    constant ADDI: std_logic_vector(4 downto 0):="00101";
    constant SUBI: std_logic_vector(4 downto 0):="00110";
    constant ANDI: std_logic_vector(4 downto 0):="00111";
    constant ORI: std_logic_vector(4 downto 0):="01000";
    constant XORI: std_logic_vector(4 downto 0):="01001";
    constant NANDI: std_logic_vector(4 downto 0):="01010";
    constant NORI: std_logic_vector(4 downto 0):="01011";
    constant XNORI: std_logic_vector(4 downto 0):="01100";
    constant BEQI: std_logic_vector(4 downto 0):="01101";
    constant BNEI: std_logic_vector(4 downto 0):="01110";
    constant BLTI: std_logic_vector(4 downto 0):="01111";
    constant BLETI: std_logic_vector(4 downto 0):="10000";
    constant BGTI: std_logic_vector(4 downto 0):="10001";
    constant BGETI: std_logic_vector(4 downto 0):="10010";
    constant B: std_logic_vector(4 downto 0):="10011";
    constant CALL: std_logic_vector(4 downto 0):="10100";
    constant RET: std_logic_vector(4 downto 0):="10101";
    constant NOP: std_logic_vector(4 downto 0):="10110";

        --REG
    CONSTANT R0: STD_LOGIC_VECTOR(3 DOWNTO 0):="0000";
    CONSTANT R1: STD_LOGIC_VECTOR(3 DOWNTO 0):="0001";
    CONSTANT R2: STD_LOGIC_VECTOR(3 DOWNTO 0):="0010";
    CONSTANT R3: STD_LOGIC_VECTOR(3 DOWNTO 0):="0011";
    CONSTANT R4: STD_LOGIC_VECTOR(3 DOWNTO 0):="0100";
    CONSTANT R5: STD_LOGIC_VECTOR(3 DOWNTO 0):="0101";
    CONSTANT R6: STD_LOGIC_VECTOR(3 DOWNTO 0):="0110";
    CONSTANT R7: STD_LOGIC_VECTOR(3 DOWNTO 0):="0111";
    CONSTANT R8: STD_LOGIC_VECTOR(3 DOWNTO 0):="1000";
```

```
    CONSTANT R9: STD_LOGIC_VECTOR(3 DOWNTO 0):="1001";
    CONSTANT R10: STD_LOGIC_VECTOR(3 DOWNTO 0):="1010";
    CONSTANT R11: STD_LOGIC_VECTOR(3 DOWNTO 0):="1011";
    CONSTANT R12: STD_LOGIC_VECTOR(3 DOWNTO 0):="1100";
    CONSTANT R13: STD_LOGIC_VECTOR(3 DOWNTO 0):="1101";
    CONSTANT R14: STD_LOGIC_VECTOR(3 DOWNTO 0):="1110";
    CONSTANT R15: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111";

        --S/U
    CONSTANT SU: std_logic_vector(3 downto 0):="0000";
        --C-FIN-OPERACION
    CONSTANT ADD: STD_LOGIC_VECTOR(3 DOWNTO 0):="0000";
    CONSTANT SUB: STD_LOGIC_VECTOR(3 DOWNTO 0):="0001";
    CONSTANT C_AND: STD_LOGIC_VECTOR(3 DOWNTO 0):="0010";
    CONSTANT C_OR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0011";
    CONSTANT C_XOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0100";
    CONSTANT C_NAND: STD_LOGIC_VECTOR(3 DOWNTO 0):="0101";
    CONSTANT C_NOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0110";
    CONSTANT C_XNOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0111";
    CONSTANT C_NOT: STD_LOGIC_VECTOR(3 DOWNTO 0):="1000";
    CONSTANT C_SLL: STD_LOGIC_VECTOR(3 DOWNTO 0):="1001";
    CONSTANT C_SRL: STD_LOGIC_VECTOR(3 DOWNTO 0):="1010";

    type banco is array (0 to (2**m)-1) of std_logic_vector(n-1 downto 0);

    constant aux : banco := (
    LI & R0 & x"0000",                  --0 LI R0, 0
    LI & R1 & x"0001",                  --1 LI R1, 1
    LI & R2 & x"0000",                  --2 LI R2, 0
    LI & R3 & x"000C",                  --3 LI R3, 12
    TYPER & R4 & R0 & R1 & SU & ADD,  --4 SUMA: ADD R4, R0, R1
    SWI & R4 & x"0048",                 --5 SWI R4, 72
    ADDI & R0 & R1 & x"000",            --6 ADDI R0, R1, 0
    ADDI & R2 & R2 & x"001",            --7 ADDI R2, R2, 1
    BNEI & R3 & R2 & x"FFC",            --8 BNEI R3, R2, SUMA
    NOP & SU & SU & SU & SU & SU,       --9 FIN: NOP
    B & SU & x"0009",                   --10 B FIN
    others => (others => '0')
    );
begin
    inst <= aux(conv_integer(dir));
end Behavioral;
```

2. Test-Bench VHDL Código

```
library ieee;
library std;
use std.textio.all;
use ieee.std_logic_TEXTIO.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity tbMemPrograma is
    generic(
        m:integer:= 10;
        n:integer:= 25
    );
end tbMemPrograma;

architecture Behavioral of tbMemPrograma is
    component memPrograma is
        port(
            dir : in STD_LOGIC_VECTOR (m-1 downto 0);
            inst : out STD_LOGIC_VECTOR (n-1 downto 0)
        );
    end component;
```

```
23      signal dir: std_logic_vector(m-1 downto 0):= "0000000000";
24      signal inst: std_logic_vector (n-1 downto 0):= "00000000000000000000000000";
25  begin
26      init:memPrograma port map(
27          dir=>dir,
28          inst=>inst
29      );
30
31      testBench:process
32          file archRes: text;
33          file archVec: text;
34
35          --outputs
36          variable lineaRes: line;
37          variable varInst: std_logic_vector(n-1 downto 0);
38          variable opcode: std_logic_vector(4 downto 0);
39          variable sec1: std_logic_vector(3 downto 0);
40          variable sec2: std_logic_vector(3 downto 0);
41          variable sec3: std_logic_vector(3 downto 0);
42          variable sec4: std_logic_vector(3 downto 0);
43          variable sec5: std_logic_vector(3 downto 0);
44
45          --inputs
46          variable lineaVec: line;
47          variable varDir: integer;
48          variable cadena: string(1 to 8);
49      begin
50          file_open(archVec,"/media/d3vcr4ck/externData/materias-Sem20_2/
        arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/memoriaPrograma/
        input.txt", read_mode);
51          file_open(archRes,"/media/d3vcr4ck/externData/materias-Sem20_2/
        arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/memoriaPrograma/
        output.txt", write_mode);
52          cadena:= "      PC";
53          write(lineaRes, cadena, right, cadena'LENGTH+1);
54          cadena:= "  OPCODE";
55          write(lineaRes, cadena, right, cadena'LENGTH+1);
56          cadena:= "19...16 ";
57          write(lineaRes, cadena, right, cadena'LENGTH+1);
58          cadena:= "15...12 ";
59          write(lineaRes, cadena, right, cadena'LENGTH+1);
60          cadena:= "11...8  ";
61          write(lineaRes, cadena, right, cadena'LENGTH+1);
62          cadena:= "7...4   ";
63          write(lineaRes, cadena, right, cadena'LENGTH+1);
64          cadena:= "3...0   ";
65          write(lineaRes, cadena, right, cadena'LENGTH+1);
66          writeline(archRes, lineaRes);
67          ciclo : for i in 0 to 70 loop
68              readline(archVec, lineaVec);
69              read(lineaVec, varDir);
70              dir<= conv_std_logic_vector(varDir,m);
71              wait for 10 ns;
72              varInst:= inst;
73              -- Separa los codigos de operacion y los bits del programa para el archivo de
        salida
74              opcode:= varInst(24 downto 20);
75              sec1:= varInst(19 downto 16);
76              sec2:= varInst(15 downto 12);
77              sec3:= varInst(11 downto 8);
78              sec4:= varInst(7 downto 4);
79              sec5:= varInst(3 downto 0);
80              write(lineaRes, varDir,right,9);
81              write(lineaRes, opcode, right, 9);
82              write(lineaRes, sec1, right, 8);
83              write(lineaRes, sec2, right, 9);
84              write(lineaRes, sec3, right, 8);
```

```
            write(lineaRes, sec4, right, 8);
            write(lineaRes, sec5, right, 9);
            writeline(archRes, lineaRes);
        end loop;
        file_close(archVec);
        file_close(archRes);
        wait;
    end process;

end Behavioral;
```

Archivo de entrada (input.txt)

```
0
1
2
3
4
5
6
7
8
4
5
6
7
8
4
5
6
7
8
4
5
6
7
8
4
5
6
7
8
4
5
6
7
8
4
5
6
7
8
4
5
6
7
8
4
5
6
7
8
4
```

```
5
6
7
8
4
5
6
7
8
9
10
9
10
9
10
```

Archivo de salida (output.txt)

```
          PC   OPCODE 19...16  15...12  11...8   7...4    3...0
           0    00001    0000     0000    0000    0000     0000
           1    00001    0001     0000    0000    0000     0001
           2    00001    0010     0000    0000    0000     0000
           3    00001    0011     0000    0000    0000     1100
           4    00000    0100     0000    0001    0000     0000
           5    00011    0100     0000    0000    0100     1000
           6    00101    0000     0001    0000    0000     0000
           7    00101    0010     0010    0000    0000     0001
           8    01110    0011     0010    1111    1111     1100
           4    00000    0100     0000    0001    0000     0000
           5    00011    0100     0000    0000    0100     1000
           6    00101    0000     0001    0000    0000     0000
           7    00101    0010     0010    0000    0000     0001
           8    01110    0011     0010    1111    1111     1100
           4    00000    0100     0000    0001    0000     0000
           5    00011    0100     0000    0000    0100     1000
           6    00101    0000     0001    0000    0000     0000
           7    00101    0010     0010    0000    0000     0001
           8    01110    0011     0010    1111    1111     1100
           4    00000    0100     0000    0001    0000     0000
           5    00011    0100     0000    0000    0100     1000
           6    00101    0000     0001    0000    0000     0000
           7    00101    0010     0010    0000    0000     0001
           8    01110    0011     0010    1111    1111     1100
           4    00000    0100     0000    0001    0000     0000
           5    00011    0100     0000    0000    0100     1000
           6    00101    0000     0001    0000    0000     0000
           7    00101    0010     0010    0000    0000     0001
           8    01110    0011     0010    1111    1111     1100
           4    00000    0100     0000    0001    0000     0000
           5    00011    0100     0000    0000    0100     1000
           6    00101    0000     0001    0000    0000     0000
           7    00101    0010     0010    0000    0000     0001
           8    01110    0011     0010    1111    1111     1100
           4    00000    0100     0000    0001    0000     0000
           5    00011    0100     0000    0000    0100     1000
           6    00101    0000     0001    0000    0000     0000
           7    00101    0010     0010    0000    0000     0001
           8    01110    0011     0010    1111    1111     1100
           4    00000    0100     0000    0001    0000     0000
           5    00011    0100     0000    0000    0100     1000
           6    00101    0000     0001    0000    0000     0000
           7    00101    0010     0010    0000    0000     0001
           8    01110    0011     0010    1111    1111     1100
           4    00000    0100     0000    0001    0000     0000
           5    00011    0100     0000    0000    0100     1000
           6    00101    0000     0001    0000    0000     0000
           7    00101    0010     0010    0000    0000     0001
           8    01110    0011     0010    1111    1111     1100
```

```
         4      00000     0100     0000     0001     0000     0000
         5      00011     0100     0000     0000     0100     1000
         6      00101     0000     0001     0000     0000     0000
         7      00101     0010     0010     0000     0000     0001
         8      01110     0011     0010     1111     1111     1100
         4      00000     0100     0000     0001     0000     0000
         5      00011     0100     0000     0000     0100     1000
         6      00101     0000     0001     0000     0000     0000
         7      00101     0010     0010     0000     0000     0001
         8      01110     0011     0010     1111     1111     1100
         4      00000     0100     0000     0001     0000     0000
         5      00011     0100     0000     0000     0100     1000
         6      00101     0000     0001     0000     0000     0000
         7      00101     0010     0010     0000     0000     0001
         8      01110     0011     0010     1111     1111     1100
         9      10110     0000     0000     0000     0000     0000
        10      10011     0000     0000     0000     0000     1001
         9      10110     0000     0000     0000     0000     0000
        10      10011     0000     0000     0000     0000     1001
         9      10110     0000     0000     0000     0000     0000
        10      10011     0000     0000     0000     0000     1001
        10      10011     0000     0000     0000     0000     1001
```

3. Simulaciones

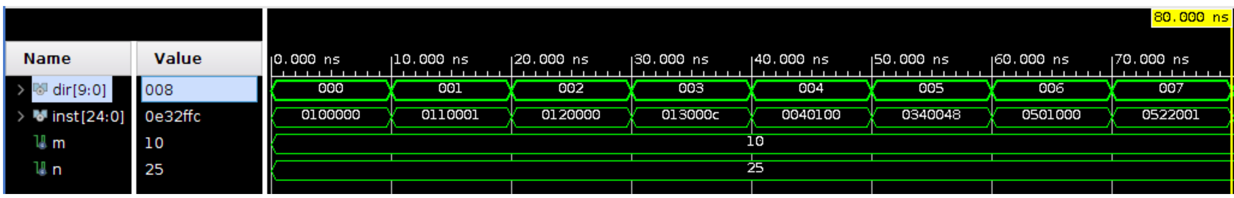

Figura 1: Primer parte del test-bench

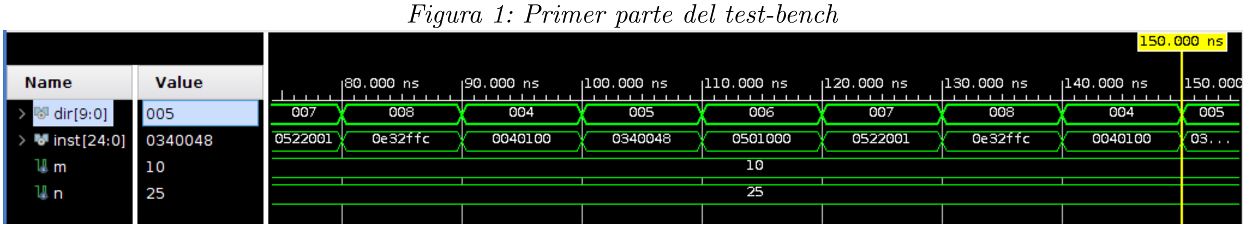

Figura 2: Segunda parte del test-bench

Figura 3: Tercer parte del test-bench

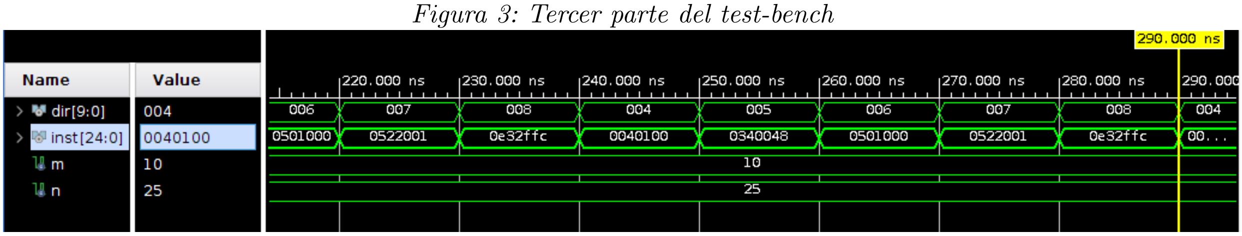

Figura 4: Cuarta parte del test-bench

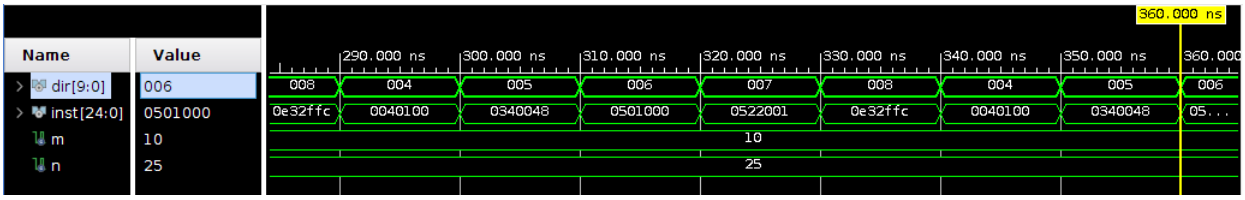

Figura 5: Quinta parte del test-bench

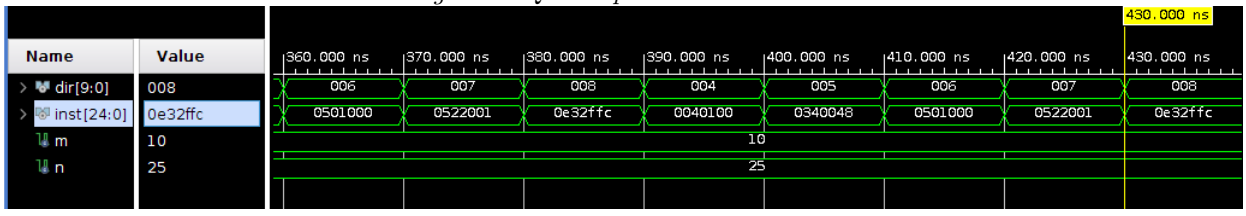

Figura 6: Sexta parte del test-bench

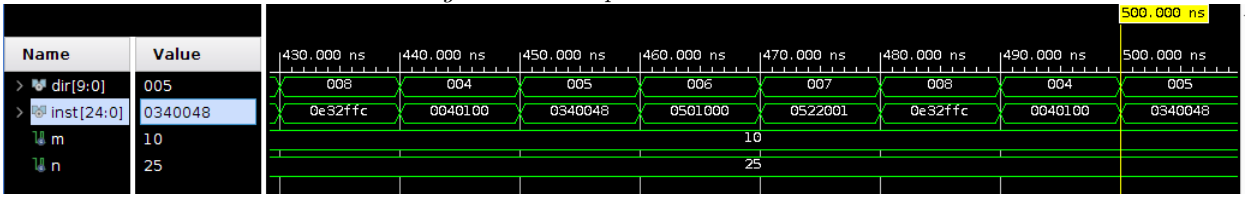

Figura 7: Septima parte del test-bench

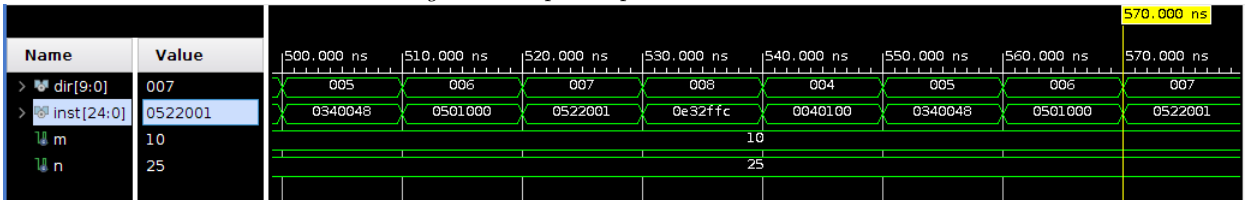

Figura 8: Octava parte del test-bench

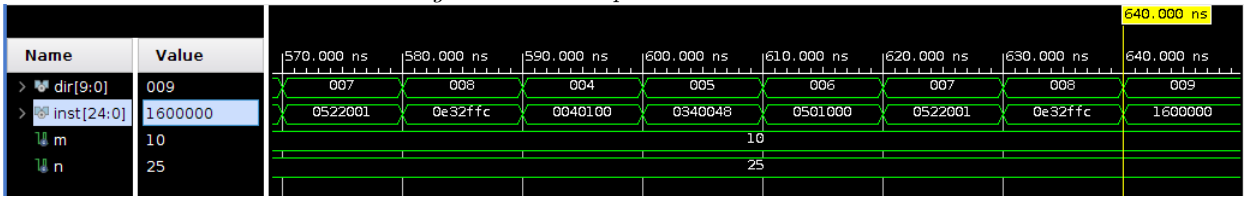

Figura 9: Novena parte del test-bench

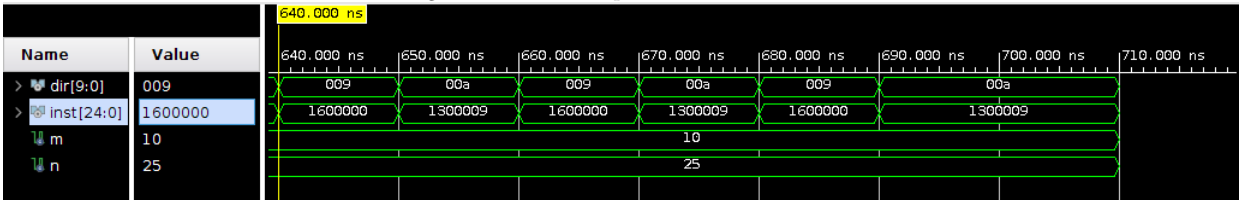

Figura 10: Decima parte del test-bench

4. Diagrama RTL