

Reporte de practica 8

González Pardo Adrian

Abril 2020

1. Código VHDL

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_arith.ALL;
4 use IEEE.STD_LOGIC_unsigned.ALL;
5
6 entity memPrograma is
7     generic(
8         m:integer:= 10;--tamano del bus de direcciones
9         n:integer:= 25--tamano de palabra
10    );
11    Port(dir:in STD_LOGIC_VECTOR (m-1 downto 0);
12          inst:out STD_LOGIC_VECTOR (n-1 downto 0));
13 end memPrograma;
14
15 architecture Behavioral of memPrograma is
16     --C-OPERACION
17     constant TYPER: std_logic_vector(4 downto 0):="00000";
18     constant LI: std_logic_vector(4 downto 0):="00001";
19     constant LWI: std_logic_vector(4 downto 0):="00010";
20     constant LW: std_logic_vector(4 downto 0):="10111";
21     constant SWI: std_logic_vector(4 downto 0):="00011";
22     constant SW: std_logic_vector(4 downto 0):="00100";
23     constant ADDI: std_logic_vector(4 downto 0):="00101";
24     constant SUBI: std_logic_vector(4 downto 0):="00110";
25     constant ANDI: std_logic_vector(4 downto 0):="00111";
26     constant ORI: std_logic_vector(4 downto 0):="01000";
27     constant XORI: std_logic_vector(4 downto 0):="01001";
28     constant NANDI: std_logic_vector(4 downto 0):="01010";
29     constant NORI: std_logic_vector(4 downto 0):="01011";
30     constant XNORI: std_logic_vector(4 downto 0):="01100";
31     constant BEQI: std_logic_vector(4 downto 0):="01101";
32     constant BNEI: std_logic_vector(4 downto 0):="01110";
33     constant BLTI: std_logic_vector(4 downto 0):="01111";
34     constant BLETI: std_logic_vector(4 downto 0):="10000";
35     constant BGTI: std_logic_vector(4 downto 0):="10001";
36     constant BGETI: std_logic_vector(4 downto 0):="10010";
37     constant B: std_logic_vector(4 downto 0):="10011";
38     constant CALL: std_logic_vector(4 downto 0):="10100";
39     constant RET: std_logic_vector(4 downto 0):="10101";
40     constant NOP: std_logic_vector(4 downto 0):="10110";
41
42     --REG
43     CONSTANT R0: STD_LOGIC_VECTOR(3 DOWNT0 0):="0000";
44     CONSTANT R1: STD_LOGIC_VECTOR(3 DOWNT0 0):="0001";
45     CONSTANT R2: STD_LOGIC_VECTOR(3 DOWNT0 0):="0010";
46     CONSTANT R3: STD_LOGIC_VECTOR(3 DOWNT0 0):="0011";
47     CONSTANT R4: STD_LOGIC_VECTOR(3 DOWNT0 0):="0100";
48     CONSTANT R5: STD_LOGIC_VECTOR(3 DOWNT0 0):="0101";
49     CONSTANT R6: STD_LOGIC_VECTOR(3 DOWNT0 0):="0110";
50     CONSTANT R7: STD_LOGIC_VECTOR(3 DOWNT0 0):="0111";
51     CONSTANT R8: STD_LOGIC_VECTOR(3 DOWNT0 0):="1000";
```

```

52  CONSTANT R9: STD_LOGIC_VECTOR(3 DOWNTO 0):="1001";
53  CONSTANT R10: STD_LOGIC_VECTOR(3 DOWNTO 0):="1010";
54  CONSTANT R11: STD_LOGIC_VECTOR(3 DOWNTO 0):="1011";
55  CONSTANT R12: STD_LOGIC_VECTOR(3 DOWNTO 0):="1100";
56  CONSTANT R13: STD_LOGIC_VECTOR(3 DOWNTO 0):="1101";
57  CONSTANT R14: STD_LOGIC_VECTOR(3 DOWNTO 0):="1110";
58  CONSTANT R15: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111";
59
60  --S/U
61  CONSTANT SU: std_logic_vector(3 downto 0):="0000";
62  --C-FIN-OPERACION
63  CONSTANT ADD: STD_LOGIC_VECTOR(3 DOWNTO 0):="0000";
64  CONSTANT SUB: STD_LOGIC_VECTOR(3 DOWNTO 0):="0001";
65  CONSTANT C_AND: STD_LOGIC_VECTOR(3 DOWNTO 0):="0010";
66  CONSTANT C_OR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0011";
67  CONSTANT C_XOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0100";
68  CONSTANT C_NAND: STD_LOGIC_VECTOR(3 DOWNTO 0):="0101";
69  CONSTANT C_NOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0110";
70  CONSTANT C_XNOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0111";
71  CONSTANT C_NOT: STD_LOGIC_VECTOR(3 DOWNTO 0):="1000";
72  CONSTANT C_SLL: STD_LOGIC_VECTOR(3 DOWNTO 0):="1001";
73  CONSTANT C_SRL: STD_LOGIC_VECTOR(3 DOWNTO 0):="1010";
74
75  type banco is array (0 to (2**m)-1) of std_logic_vector(n-1 downto 0);
76
77  constant aux : banco := (
78    LI & R0 & x"0000",          --0 LI R0, 0
79    LI & R1 & x"0001",          --1 LI R1, 1
80    LI & R2 & x"0000",          --2 LI R2, 0
81    LI & R3 & x"000C",          --3 LI R3, 12
82    TYPER & R4 & R0 & R1 & SU & ADD, --4 SUMA: ADD R4, R0, R1
83    SWI & R4 & x"0048",          --5 SWI R4, 72
84    ADDI & R0 & R1 & x"000",      --6 ADDI R0, R1, 0
85    ADDI & R2 & R2 & x"001",      --7 ADDI R2, R2, 1
86    BNEI & R3 & R2 & x"FFC",      --8 BNEI R3, R2, SUMA
87    NOP & SU & SU & SU & SU & SU, --9 FIN: NOP
88    B & SU & x"0009",            --10 B FIN
89    others => (others => '0')
90  );
91  begin
92    inst <= aux(conv_integer(dir));
93  end Behavioral;

```

2. Test-Bench VHDL Código

```

1  library ieee;
2  library std;
3  use std.textio.all;
4  use ieee.std_logic_TEXTIO.all;
5  use ieee.std_logic_1164.all;
6  use ieee.std_logic_unsigned.all;
7  use ieee.std_logic_arith.all;
8
9  entity tbMemPrograma is
10     generic(
11         m:integer:= 10;
12         n:integer:= 25
13     );
14  end tbMemPrograma;
15
16  architecture Behavioral of tbMemPrograma is
17     component memPrograma is
18         port(
19             dir : in STD_LOGIC_VECTOR (m-1 downto 0);
20             inst : out STD_LOGIC_VECTOR (n-1 downto 0)
21         );
22     end component;

```

```

23     signal dir: std_logic_vector(m-1 downto 0) := "0000000000";
24     signal inst: std_logic_vector (n-1 downto 0) := "000000000000000000000000";
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);

```

```

85         write(lineaRes, sec4, right, 8);
86         write(lineaRes, sec5, right, 9);
87         writeline(archRes, lineaRes);
88     end loop;
89     file_close(archVec);
90     file_close(archRes);
91     wait;
92 end process;
93
94 end Behavioral;

```

Archivo de entrada (input.txt)

```

1 0
2 1
3 2
4 3
5 4
6 5
7 6
8 7
9 8
10 4
11 5
12 6
13 7
14 8
15 4
16 5
17 6
18 7
19 8
20 4
21 5
22 6
23 7
24 8
25 4
26 5
27 6
28 7
29 8
30 4
31 5
32 6
33 7
34 8
35 4
36 5
37 6
38 7
39 8
40 4
41 5
42 6
43 7
44 8
45 4
46 5
47 6
48 7
49 8
50 4
51 5
52 6
53 7
54 8
55 4

```

56 5
57 6
58 7
59 8
60 4
61 5
62 6
63 7
64 8
65 9
66 10
67 9
68 10
69 9
70 10

Archivo de salida (output.txt)

	PC	OPCODE	19...16	15...12	11...8	7...4	3...0
1	0	00001	0000	0000	0000	0000	0000
2	1	00001	0001	0000	0000	0000	0001
3	2	00001	0010	0000	0000	0000	0000
4	3	00001	0011	0000	0000	0000	1100
5	4	00000	0100	0000	0001	0000	0000
6	5	00011	0100	0000	0000	0100	1000
7	6	00101	0000	0001	0000	0000	0000
8	7	00101	0010	0010	0000	0000	0001
9	8	01110	0011	0010	1111	1111	1100
10	4	00000	0100	0000	0001	0000	0000
11	5	00011	0100	0000	0000	0100	1000
12	6	00101	0000	0001	0000	0000	0000
13	7	00101	0010	0010	0000	0000	0001
14	8	01110	0011	0010	1111	1111	1100
15	4	00000	0100	0000	0001	0000	0000
16	5	00011	0100	0000	0000	0100	1000
17	6	00101	0000	0001	0000	0000	0000
18	7	00101	0010	0010	0000	0000	0001
19	8	01110	0011	0010	1111	1111	1100
20	4	00000	0100	0000	0001	0000	0000
21	5	00011	0100	0000	0000	0100	1000
22	6	00101	0000	0001	0000	0000	0000
23	7	00101	0010	0010	0000	0000	0001
24	8	01110	0011	0010	1111	1111	1100
25	4	00000	0100	0000	0001	0000	0000
26	5	00011	0100	0000	0000	0100	1000
27	6	00101	0000	0001	0000	0000	0000
28	7	00101	0010	0010	0000	0000	0001
29	8	01110	0011	0010	1111	1111	1100
30	4	00000	0100	0000	0001	0000	0000
31	5	00011	0100	0000	0000	0100	1000
32	6	00101	0000	0001	0000	0000	0000
33	7	00101	0010	0010	0000	0000	0001
34	8	01110	0011	0010	1111	1111	1100
35	4	00000	0100	0000	0001	0000	0000
36	5	00011	0100	0000	0000	0100	1000
37	6	00101	0000	0001	0000	0000	0000
38	7	00101	0010	0010	0000	0000	0001
39	8	01110	0011	0010	1111	1111	1100
40	4	00000	0100	0000	0001	0000	0000
41	5	00011	0100	0000	0000	0100	1000
42	6	00101	0000	0001	0000	0000	0000
43	7	00101	0010	0010	0000	0000	0001
44	8	01110	0011	0010	1111	1111	1100
45	4	00000	0100	0000	0001	0000	0000
46	5	00011	0100	0000	0000	0100	1000
47	6	00101	0000	0001	0000	0000	0000
48	7	00101	0010	0010	0000	0000	0001
49	8	01110	0011	0010	1111	1111	1100

51	4	00000	0100	0000	0001	0000	0000
52	5	00011	0100	0000	0000	0100	1000
53	6	00101	0000	0001	0000	0000	0000
54	7	00101	0010	0010	0000	0000	0001
55	8	01110	0011	0010	1111	1111	1100
56	4	00000	0100	0000	0001	0000	0000
57	5	00011	0100	0000	0000	0100	1000
58	6	00101	0000	0001	0000	0000	0000
59	7	00101	0010	0010	0000	0000	0001
60	8	01110	0011	0010	1111	1111	1100
61	4	00000	0100	0000	0001	0000	0000
62	5	00011	0100	0000	0000	0100	1000
63	6	00101	0000	0001	0000	0000	0000
64	7	00101	0010	0010	0000	0000	0001
65	8	01110	0011	0010	1111	1111	1100
66	9	10110	0000	0000	0000	0000	0000
67	10	10011	0000	0000	0000	0000	1001
68	9	10110	0000	0000	0000	0000	0000
69	10	10011	0000	0000	0000	0000	1001
70	9	10110	0000	0000	0000	0000	0000
71	10	10011	0000	0000	0000	0000	1001
72	10	10011	0000	0000	0000	0000	1001

3. Simulaciones

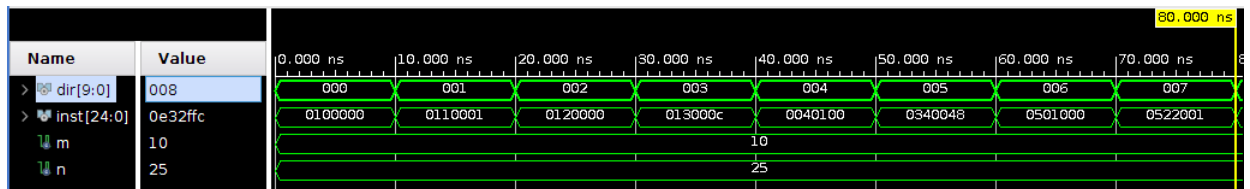


Figura 1: Primer parte del test-bench

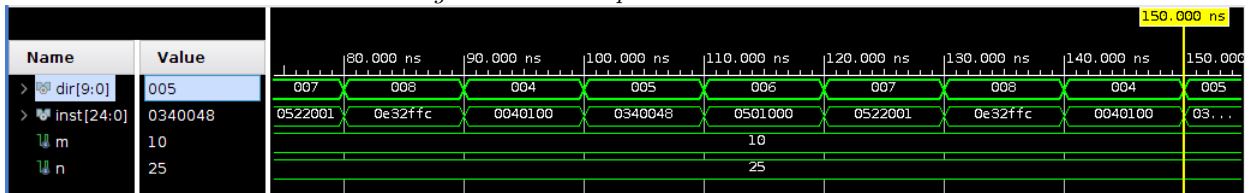


Figura 2: Segunda parte del test-bench

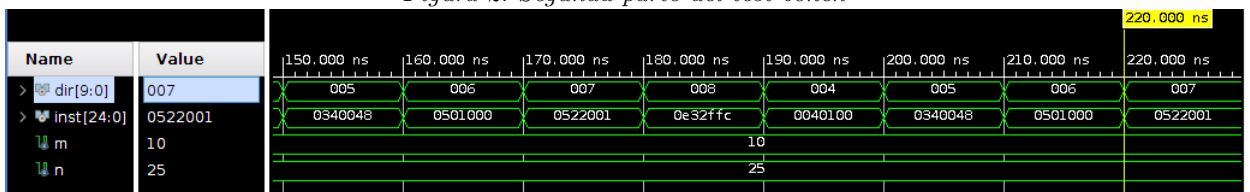


Figura 3: Tercer parte del test-bench

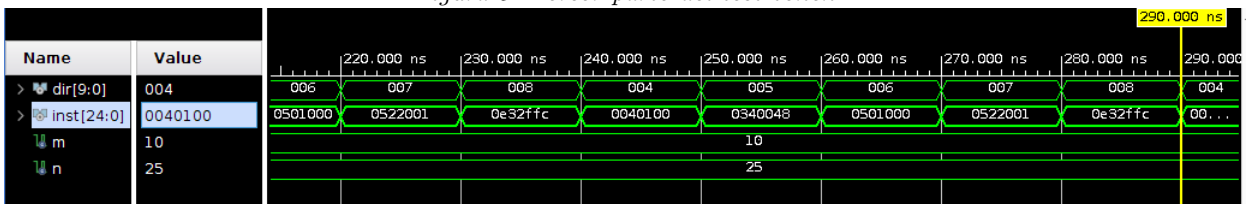


Figura 4: Cuarta parte del test-bench

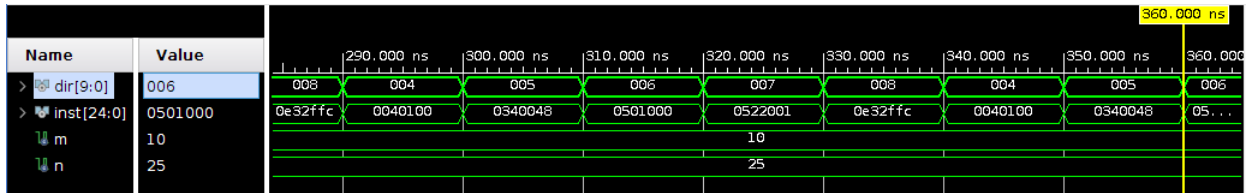


Figura 5: Quinta parte del test-bench

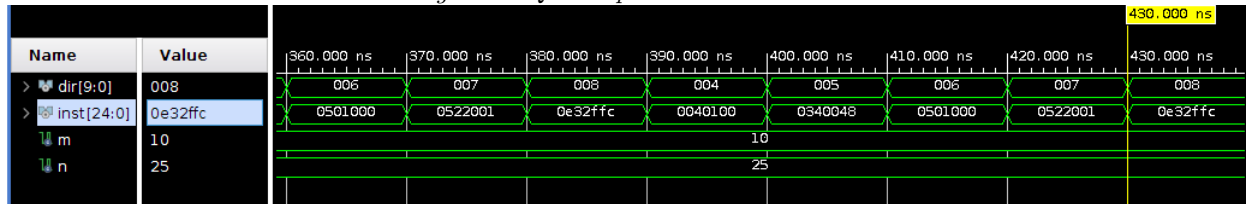


Figura 6: Sexta parte del test-bench

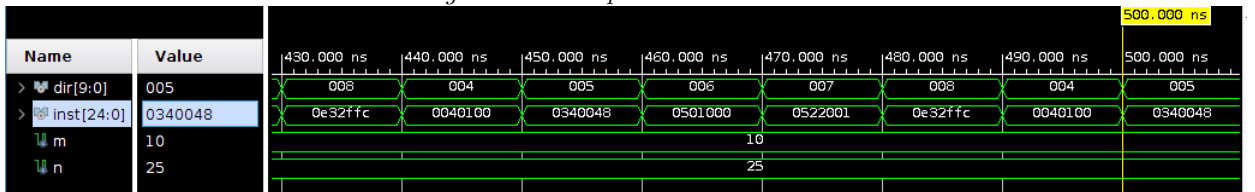


Figura 7: Septima parte del test-bench

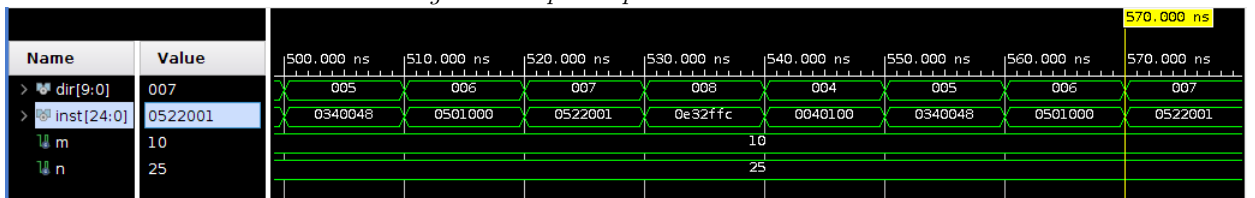


Figura 8: Octava parte del test-bench

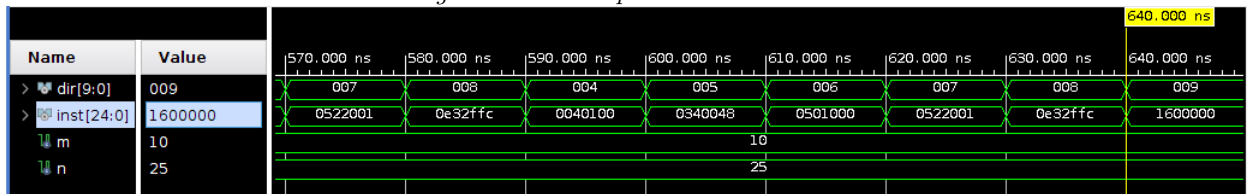


Figura 9: Novena parte del test-bench

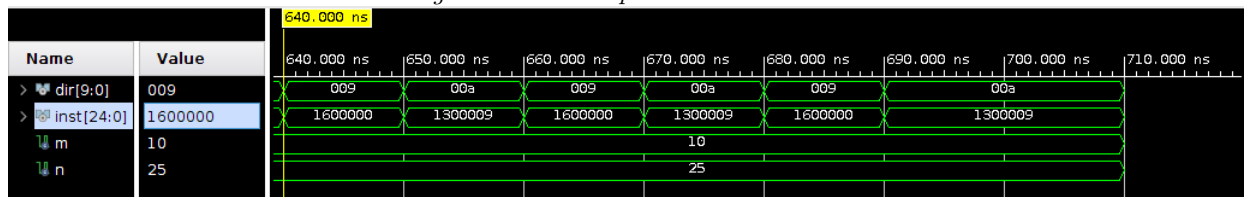


Figura 10: Decima parte del test-bench

4. Diagrama RTL

