Martínez Coronel Brayan Yosafat

CÓDIGO VHDL

MEMORIA DE DATOS

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

entity memoria is
    generic (
        m : integer := 11;
        n : integer := 16
    );
    Port ( add : in STD_LOGIC_VECTOR (m-1 downto 0);
           dataIn : in STD_LOGIC_VECTOR (n-1 downto 0);
           clk, WD : in STD_LOGIC;
           dataOut : out STD_LOGIC_VECTOR (n-1 downto 0)
           );
end entity;

architecture Behavioral of memoria 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
    process (clk)
    begin
        if (rising_edge(clk)) then
            if (WD = '1') then
                mem(conv_integer(add)) <= dataIn;
            else
                mem <= mem;
            end if;
        end if;
    end process;

    dataOut <= mem(conv_integer(add));
end Behavioral;
```

MEMORIA DE PROGRAMA

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

entity memoria is
    generic (
        m : integer := 10;
        n : integer := 25
    );
    Port ( pc : in STD_LOGIC_VECTOR (m-1 downto 0);
           inst : out STD_LOGIC_VECTOR (n-1 downto 0)
           );
end entity;

architecture Behavioral of memoria is
type arreglo is array (0 to (2**m - 1)) of STD_LOGIC_VECTOR (n-1 downto 0);
signal mem : arreglo := (others=>(others=>'0'));

constant code_LI : std_logic_vector (4 downto 0) := "00001";
constant code_ADD : std_logic_vector (4 downto 0) := "00000";
constant code_SWI : std_logic_vector (4 downto 0) := "00011";
constant code_ADDI : std_logic_vector (4 downto 0) := "00101";
constant code_BNEI : std_logic_vector (4 downto 0) := "01110";
constant code_NOP : std_logic_vector (4 downto 0) := "10110";

begin
    mem(0) <= code_LI & "0000" & "000000000000" & "0000";
    mem(1) <= code_LI & "0001" & "000000000000" & "0001";
    mem(2) <= code_LI & "0010" & "000000000000" & "0000";
    mem(3) <= code_LI & "0011" & "000000000000" & "1100";
    mem(4) <= code_ADD & "0100" & "0000" & "0001" & "00000000";
    mem(5) <= code_SWI & "0100" & "00000000" & "0111" & "0010";
    mem(6) <= code_ADDI & "0000" & "0001" & "000000000000";
    mem(7) <= code_ADDI & "0001" & "0100" & "000000000000";
    mem(8) <= code_ADDI & "0010" & "0010" & "000000000001";
    mem(9) <= code_BNEI & "0011" & "0010" & "111111111011";

    mem(10) <= code_NOP & "00000000000000000000";

    inst <= mem(conv_integer(pc));
end Behavioral;
```

CÓDIGO DEL TEST-BENCH

MEMORIA DE DATOS

```
library IEEE;
library STD;
use STD.TEXTIO.ALL;
use IEEE.std_logic_TEXTIO.ALL;
use IEEE.STD_LOGIC_1164.ALL;

entity t_memoria is
--  Port ( );
end t_memoria;

architecture Behavioral of t_memoria is
constant m : integer := 11;
constant n : integer := 16;
constant clk_period : time := 10 ns;

component memoria is generic (
        m : integer := m;
        n : integer := n
    );
    Port ( add : in STD_LOGIC_VECTOR (m-1 downto 0);
       dataIn : in STD_LOGIC_VECTOR (n-1 downto 0);
       clk, WD : in STD_LOGIC;
       dataOut : out STD_LOGIC_VECTOR (n-1 downto 0)
    );
end component;

signal add : STD_LOGIC_VECTOR (m-1 downto 0);
signal dataIn : STD_LOGIC_VECTOR (n-1 downto 0);
signal clk, WD : STD_LOGIC;
signal dataOut : STD_LOGIC_VECTOR (n-1 downto 0);
begin
    test: memoria port map (
        add => add,
        dataIn => dataIn,
        clk => clk,
        WD => WD,
        dataOut => dataOut
    );
```

```
  clk_process : process
  begin
                      CLK <= '0';
                        wait for clk_period/2;
                      CLK <= '1';
                        wait for clk_period/2;
  end process;

   stim_proc: process
   file ARCH_SALIDA : TEXT;

   variable LINEA_SALIDA : line;

   file ARCH_ENTRADA : TEXT;
   variable LINEA_ENTRADA : line;
   variable CADENA : string (1 to 4);
   variable v_add : STD_LOGIC_VECTOR (m-1 downto 0);
   variable v_dataIn : STD_LOGIC_VECTOR (n-1 downto 0);
   variable v_WD : STD_LOGIC;
   variable v_dataOut : STD_LOGIC_VECTOR (n-1 downto 0);
   begin
      file_open(ARCH_ENTRADA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica6\entradas1.txt",
READ_MODE);
      file_open(ARCH_SALIDA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica6\salidas1.txt",
WRITE_MODE);

      CADENA := " ADD";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "  WD";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := " DIN";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "DOUT";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);

      writeline(ARCH_SALIDA, LINEA_SALIDA);-- escribe la linea en el archivo

      WAIT FOR 100 NS;
      FOR I IN 0 TO 11 LOOP
         readline(ARCH_ENTRADA, LINEA_ENTRADA); -- lee una linea completa

         Hread(LINEA_ENTRADA, v_add);
         add <= v_add;
```

```
            Hread(LINEA_ENTRADA, v_dataIn);
            dataIn <= v_dataIn;
            read(LINEA_ENTRADA, v_WD);
            WD <= v_WD;

            WAIT UNTIL RISING_EDGE(clk);       --ESPERO AL FLANCO DE SUBIDA

            v_dataOut := dataOut;

            Hwrite(LINEA_SALIDA, v_add, right, 5);
            write(LINEA_SALIDA, v_WD, right, 5);
            Hwrite(LINEA_SALIDA, v_dataIn, right, 5);
            Hwrite(LINEA_SALIDA, v_dataOut, right, 5);

            writeline(ARCH_SALIDA, LINEA_SALIDA);-- escribe la linea en el archivo

        end loop;
        file_close(ARCH_ENTRADA);  -- cierra el archivo
        file_close(ARCH_SALIDA);  -- cierra el archivo

      wait;
    end process;
end Behavioral;
```

MEMORIA DE PROGRAMA

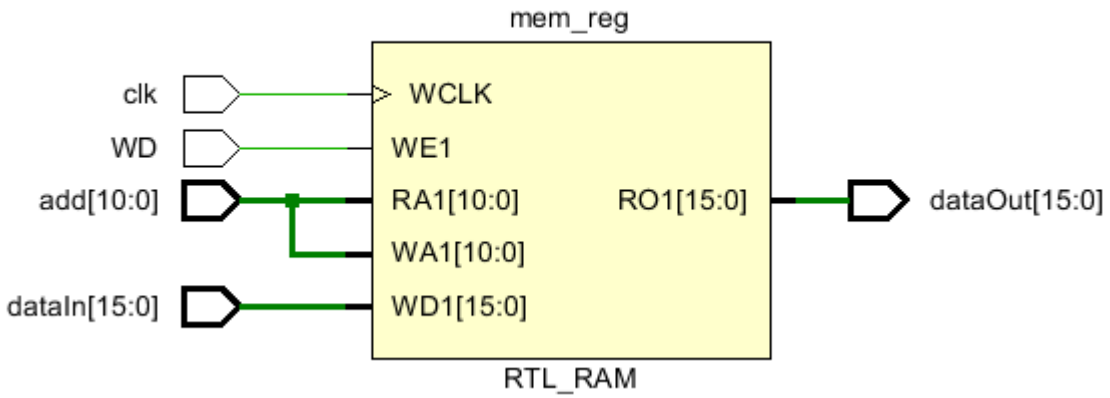
```
library IEEE;
library STD;
use STD.TEXTIO.ALL;
use IEEE.std_logic_TEXTIO.ALL;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity t_memoria is
--  Port ( );
end t_memoria;

architecture Behavioral of t_memoria is
constant m : integer := 10;
constant n : integer := 25;
constant period : time := 20 ns;

component memoria is generic (
        m : integer := m;
        n : integer := n
    );
    Port (
        pc : in STD_LOGIC_VECTOR (m-1 downto 0);
        inst : out STD_LOGIC_VECTOR (n-1 downto 0)
    );
end component;

signal pc : STD_LOGIC_VECTOR (m-1 downto 0) := (others => '0');
signal inst : STD_LOGIC_VECTOR (n-1 downto 0);
begin
    test: memoria port map (
        pc => pc,
        inst => inst
    );

    stim_proc: process
    file ARCH_SALIDA : TEXT;

    variable LINEA_SALIDA : line;
    variable v_inst : STD_LOGIC_VECTOR (n-1 downto 0);

    file ARCH_ENTRADA : TEXT;
```

```
    variable LINEA_ENTRADA : line;
    variable CADENA : string (1 to 7);
    variable v_pc : STD_LOGIC_VECTOR (m-1 downto 0);
    begin
        file_open(ARCH_ENTRADA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica6\entradas2.txt",
READ_MODE);
        file_open(ARCH_SALIDA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica6\salidas2.txt",
WRITE_MODE);

        CADENA := "     PC";
        write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
        CADENA := " OPCODE";
        write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
        CADENA := " 19..16";
        write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
        CADENA := " 15..12";
        write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
        CADENA := "  11..8";
        write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
        CADENA := "   7..4";
        write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
        CADENA := "   3..0";
        write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);


        writeline(ARCH_SALIDA, LINEA_SALIDA);-- escribe la linea en el archivo

        WAIT FOR 100 NS;
        FOR I IN 0 TO 9 LOOP
            readline(ARCH_ENTRADA, LINEA_ENTRADA); -- lee una linea completa

            Hread(LINEA_ENTRADA, v_pc);
            pc <= v_pc;

            wait for 20 ns;

            v_inst := inst;

            Hwrite(LINEA_SALIDA, v_pc, right, 9);
            write(LINEA_SALIDA, v_inst(24 downto 20), right, 7);
            write(LINEA_SALIDA, v_inst(19 downto 16), right, 7);
            write(LINEA_SALIDA, v_inst(15 downto 12), right, 8);
            write(LINEA_SALIDA, v_inst(11 downto 8), right, 9);
```

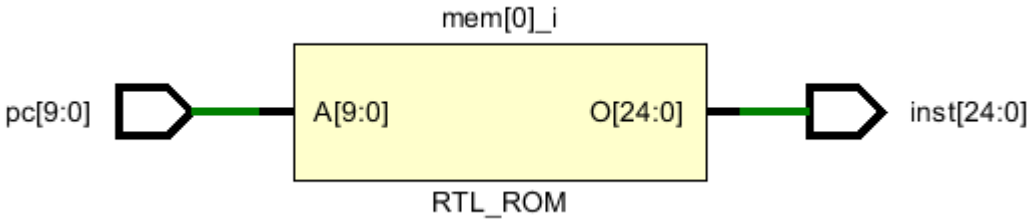
```
            write(LINEA_SALIDA, v_inst(7 downto 4), right, 8);
            write(LINEA_SALIDA, v_inst(3 downto 0), right, 8);

            writeline(ARCH_SALIDA, LINEA_SALIDA);-- escribe la linea en el archivo

        end loop;
        file_close(ARCH_ENTRADA);  -- cierra el archivo
        file_close(ARCH_SALIDA);  -- cierra el archivo

      wait;
    end process;
end Behavioral;
```

DIAGRAMAS RTL

MEMORIA DE DATOS

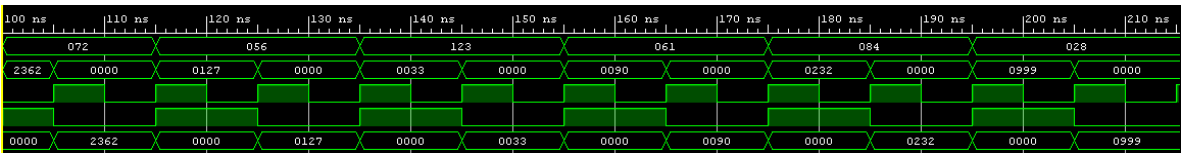


MEMORIA DE PROGRAMA

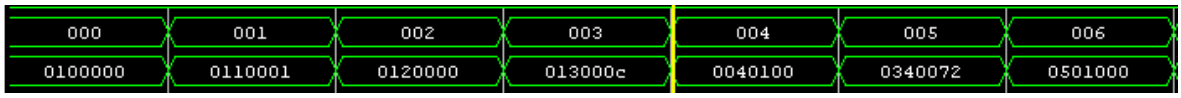


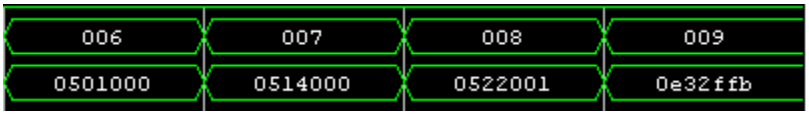
FORMA DE ONDA

MEMORIA DE DATOS



MEMORIA DE PROGRAMA





ARCHIVOS

ENTRADA – MEMORIA DE DATOS

```
072 2362 1
072 0000 0
056 0127 1
056 0000 0
123 0033 1
123 0000 0
061 0090 1
061 0000 0
084 0232 1
084 0000 0
028 0999 1
028 0000 0
```

SALIDA – MEMORIA DE DATOS

ADD	WD	DIN	DOUT
072	1	2362	0000
072	0	0000	2362
056	1	0127	0000
056	0	0000	0127
123	1	0033	0000
123	0	0000	0033
061	1	0090	0000
061	0	0000	0090
084	1	0232	0000
084	0	0000	0232
028	1	0999	0000
028	0	0000	0999

ENTRADA – MEMORIA DE PROGRAMA

```
000
001
002
003
004
005
006
007
008
009
00A
```

SALIDA - MEMORIA DE PROGRAMA

PC	OPCODE	19..16	15..12	11..8	7..4	3..0
000	00001	0000	0000	0000	0000	0000
001	00001	0001	0000	0000	0000	0001
002	00001	0010	0000	0000	0000	0000
003	00001	0011	0000	0000	0000	1100
004	00000	0100	0000	0001	0000	0000
005	00011	0100	0000	0000	0111	0010
006	00101	0000	0001	0000	0000	0000
007	00101	0001	0100	0000	0000	0000
008	00101	0010	0010	0000	0000	0001
009	01110	0011	0010	1111	1111	1011