Mendoza Martínez Eduardo
3CV11

Memoria de datos

a. Código de Implementación

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use ieee.numeric_std.all;

entity memdatos is

    generic (
        p : integer := 11;
        d : integer := 16
    );
    port (
        add : in std_logic_vector(p-1 downto 0);
        datain : in std_logic_vector(d-1 downto 0);
        clk, wd : in std_logic;
        dataout : out std_logic_vector(d-1 downto 0)
    );

end memdatos;

architecture Behavioral of memdatos is

    type aux is array (0 to (2**p) - 1) of std_logic_vector(d-1 downto 0);
    signal matrix : aux;

begin

    mem: process(clk)
    begin
        if rising_edge(clk) then
            if wd = '1' then
                matrix(conv_integer(add)) <= datain;
            else
                dataout <= matrix(conv_integer(add));
            end if;
        end if;
    end process mem;
```

```
end Behavioral;
```

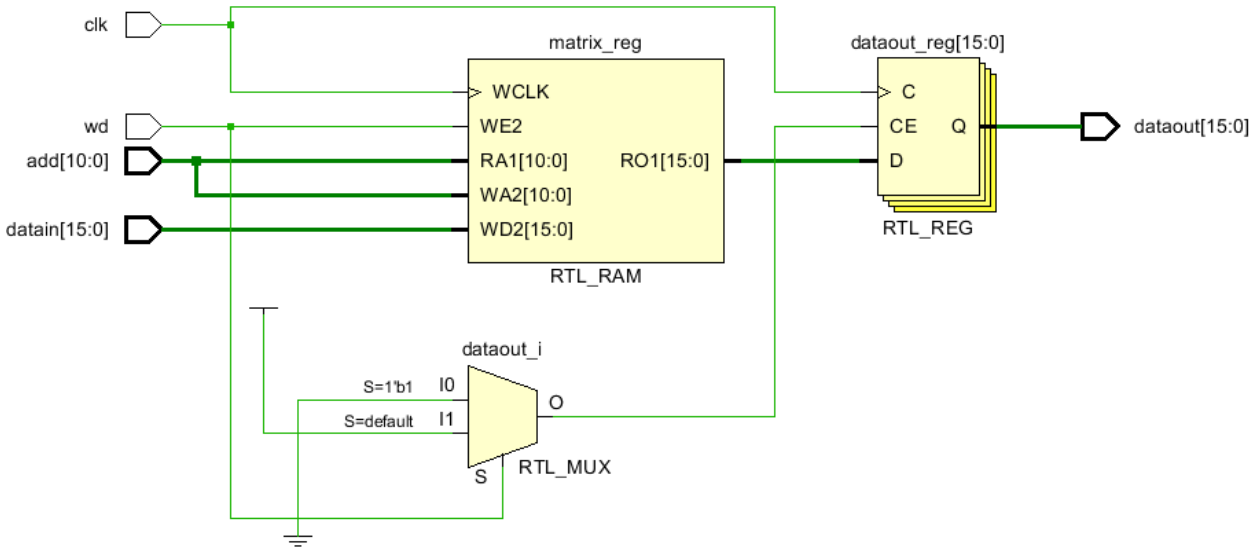
b. Código de simulación

```
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 tb_memdatos is
end tb_memdatos;

architecture Behavioral of tb_memdatos is

    component memdatos is

        generic (
            p : integer := 11;
            d : integer := 16
        );
        port (
            add : in std_logic_vector(p-1 downto 0);
            datain : in std_logic_vector(d-1 downto 0);
            clk, wd : in std_logic;
            dataout : out std_logic_vector(d-1 downto 0)
        );

    end component;

    signal add : std_logic_vector(10 downto 0) := (others => '0');
    signal datain : std_logic_vector(15 downto 0) := (others => '0');
    signal clk, wd : std_logic := '0';

    signal dataout : std_logic_vector(15 downto 0);

    constant clk_period : TIME := 50ns;

begin
```

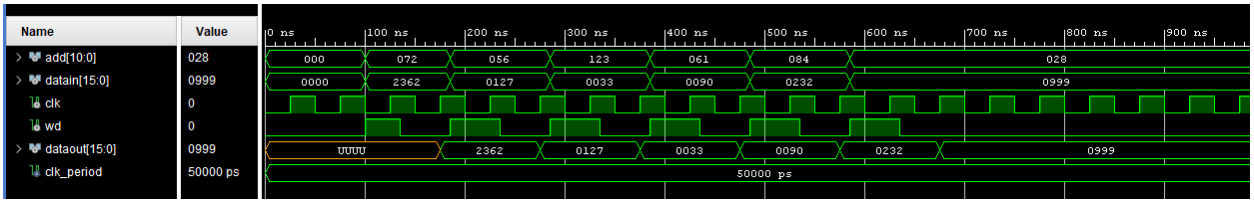
```
    uut : memdatos Port map(
        wd => wd,
        clk => clk,
        add => add,
        datain => datain,
        dataout => dataout
    );

    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_res : TEXT;
        variable linea_res : LINE;

        file arch_vec : TEXT;
        variable linea_vec : LINE;
        variable var_datain : STD_LOGIC_VECTOR(15 downto 0);
        variable var_dataout : STD_LOGIC_VECTOR(15 downto 0);
        variable var_add : STD_LOGIC_VECTOR(10 downto 0);
        variable var_wd : STD_LOGIC;
        variable cadena : STRING(1 to 7);
    begin
        file_open(arch_res,
"D:\Documents\PracticasArqui\arquitectura\individuales\P6\res.txt",
WRITE_MODE);
        file_open(arch_vec,
"D:\Documents\PracticasArqui\arquitectura\individuales\P6\vect.txt",
READ_MODE);

        cadena := "  add  ";
        write(linea_res, cadena, right, cadena'LENGTH + 1);
        cadena := " wd    ";
        write(linea_res, cadena, right, cadena'LENGTH + 1);
        cadena := " dataIn";
        write(linea_res, cadena, right, cadena'LENGTH + 1);
        cadena := "dataOut";
```

```
        write(linea_res, cadena, right, cadena'LENGTH + 1);
        writeline(arch_res, linea_res);

        wait for 100 ns;

        for i in 0 to 11 loop
            readline(arch_vec, linea_vec);

            Hread(linea_vec, var_add);
            add <= var_add;
            read(linea_vec, var_wd);
            wd <= var_wd;
            Hread(linea_vec, var_datain);
            datain <= var_datain;

            wait until rising_edge(clk);
            wait for 10ns;

            var_dataout := dataout;
            Hwrite(linea_res, var_add, right, 5);
            write(linea_res, var_wd, right, 5);
            Hwrite(linea_res, var_datain, right, 5);
            Hwrite(linea_res, var_dataout, right, 5);

            writeline(arch_res, linea_res);
        end loop;
        file_close(arch_res);
        file_close(arch_vec);

        wait;

    end process stim_proc;

end Behavioral;
```

c. Diagrama RTL



d. Forma de onda de simulación



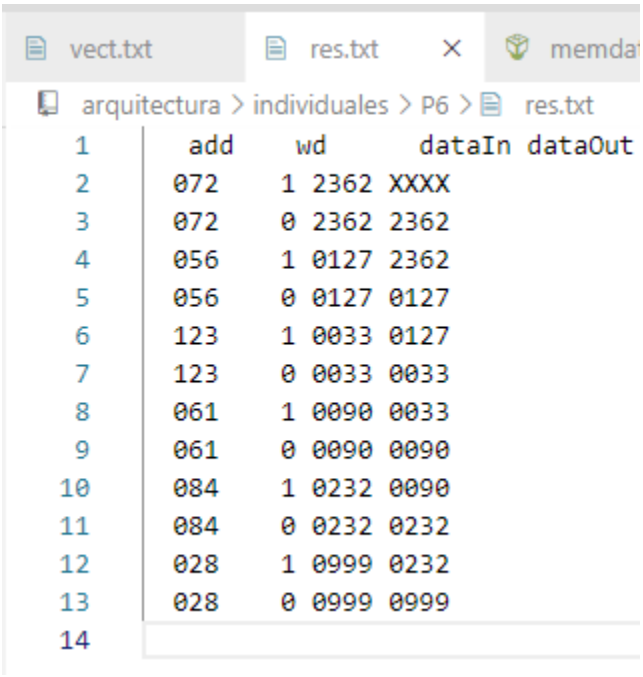
e. Captura del archivo de estímulos de entrada

vect.txt res.txt

arquitectura > individuales > P6 >

```
1   072 1 2362
2   072 0 2362
3   056 1 0127
4   056 0 0127
5   123 1 0033
6   123 0 0033
7   061 1 0090
8   061 0 0090
9   084 1 0232
10  084 0 0232
11  028 1 0999
12  028 0 0999
13
14  add w dtin
```

f. Captura del archivo de salida



Memoria de programa

a. Código de Implementación

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

entity memprog is

    generic (
        p : integer := 10;
        d : integer := 25
    );
    Port (
        PC   : in STD_LOGIC_VECTOR (p-1 downto 0);
        inst : out STD_LOGIC_VECTOR (d-1 downto 0)
    );

end memprog;

architecture Behavioral of memprog is
```

```
    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 OP_LI     : std_logic_vector(4 downto 0) := "00001";
    constant OP_LWI    : std_logic_vector(4 downto 0) := "00010";
    constant OP_LW     : std_logic_vector(4 downto 0) := "10111";
    constant OP_SWI    : std_logic_vector(4 downto 0) := "00011";
    constant OP_SW     : std_logic_vector(4 downto 0) := "00100";
    constant OP_ADD    : std_logic_vector(4 downto 0) := "00000";
    constant OP_ADDI   : std_logic_vector(4 downto 0) := "00101";
    constant OP_SUBI   : std_logic_vector(4 downto 0) := "00110";
    constant OP_BNEI   : std_logic_vector(4 downto 0) := "01110";
    constant OP_B      : std_logic_vector(4 downto 0) := "10011";
    constant OP_NOP    : std_logic_vector(4 downto 0) := "10110";

    constant SU : std_logic_vector(3 downto 0) := "0000";

    type mem is array (0 to (2**p)-1) of std_logic_vector(d-1 downto 0);
    constant aux : mem := (
        OP_LI    & R0 &      x"0000",
        OP_LI    & R1 &      x"0001",
        OP_LI    & R2 &      x"0000",
        OP_LI    & R3 &      x"000C",
        OP_ADD   & R4 & R0 & R1 & SU & SU,
        OP_SWI   & R4 &      x"0048",
        OP_ADDI  & R0 & R1 & x"000",
        OP_ADDI  & R1 & R4 & x"000",
        OP_ADDI  & R2 & R2 & x"001",
        OP_BNEI  & R3 & R2 & x"004",
        OP_NOP   & SU & SU & SU & SU & SU,
        OP_B     & SU &      x"000A",
        others => (others => '0')
    );

begin
    inst <= aux(conv_integer(PC));
end Behavioral;
```

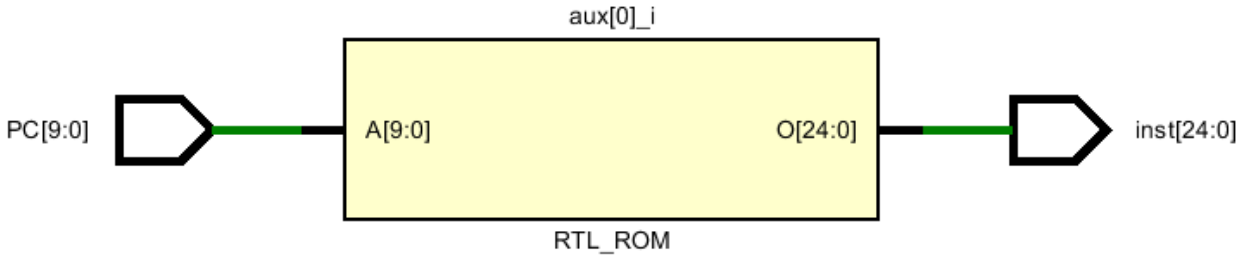
b. Código de simulación

```
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 tb_memprog is
end tb_memprog;

architecture Behavioral of tb_memprog is

    component memprog is

        generic (
            p : integer := 10;
            d : integer := 25
        );
        Port (
            PC   : in STD_LOGIC_VECTOR (p-1 downto 0);
            inst : out STD_LOGIC_VECTOR (d-1 downto 0)
        );

    end component;

    signal PC   : STD_LOGIC_VECTOR (9 downto 0) := (others => '0');
    signal inst : STD_LOGIC_VECTOR (24 downto 0);

begin

    uut: memprog port map (
        PC => PC,
        inst => inst
    );

    stim_proc: process
        file arch_res : TEXT;
        variable linea_res : LINE;
        variable var_pc : STD_LOGIC_VECTOR(9 downto 0);
```

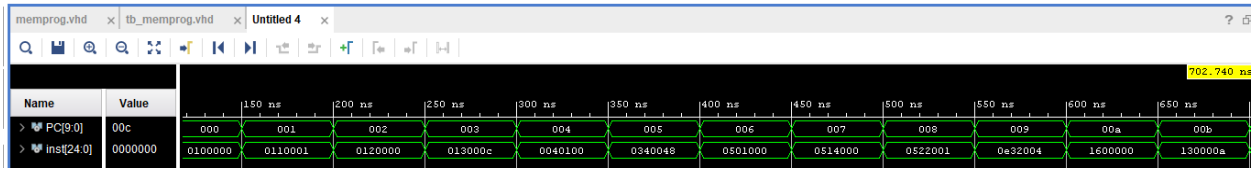
```
        variable var_inst : STD_LOGIC_VECTOR(24 downto 0);
        variable cadena : STRING(1 to 8);
    begin
        file_open(arch_res,
"D:\Documents\PracticasArqui\arquitectura\individuales\P6\prog\res.txt",
WRITE_MODE);

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

        wait for 100 ns;

        for i in 0 to 15 loop

            PC <= conv_std_logic_vector(i, 10);
            var_pc := PC;
            var_inst := inst;

            wait for 50 ns;

            Hwrite(linea_res, var_pc, right, 9);
            write(linea_res, var_inst, right, 26);

            writeline(arch_res, linea_res);
        end loop;
        file_close(arch_res);

        wait;
    end process stim_proc;
```

```
end Behavioral;
```

c. Diagrama RTL



d. Forma de onda de simulación



f. Captura del archivo de salida