Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 7 - Memoria de Datos

Unidad de aprendizaje: Arquitectura de Computadoras

Grupo: 3CV1

Alumno(a):
Ramos Diaz Enrique

Profesor(a):
Vega García Nayeli

9 de abril 2020

Índice

1. Cálculo del tamaño de los buses de datos y de direcciones

Se sabe que 2^m x n = 4096 bytes = 32768 bits

El tamaño de la palabra se obtiene de instrucciones del set ESCOMips como **LI Rd, #Slit16** con ayuda del tamaño de la literal, siendo de **n = 16 bits**.

Para obtener m es necesario despejar la primera ecuación:

$m = \log^2 \frac{(4096)(8)}{n}$

$m = \log^2 \frac{32768}{16} = 11$

m = 11 bits

2. Código de implementación

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

entity MemoriaDatos 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 MemoriaDatos;

architecture Behavioral of MemoriaDatos is
    type banco is array (0 to (2**m)-1) of STD_LOGIC_VECTOR (n-1 downto 0);
    signal aux : banco;
begin
    process(clk)
    begin
        if (rising_edge(clk)) then
            if (wd = '1') then
                aux(conv_integer(add)) <= dataIn;
            end if;
        end if;
    end process;
    dataOut <= aux(conv_integer(add));
end Behavioral;
```

3. Código de simulación

```
library ieee;
library STD;
use ieee.STD_LOGIC_1164.ALL;
use ieee.STD_LOGIC_arith.all;
use ieee.STD_LOGIC_unsigned.ALL;
use ieee.STD_LOGIC_TEXTIO.ALL;
use STD.TEXTIO.ALL;

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is
    component MemoriaDatos is
        Port ( add : in STD_LOGIC_VECTOR (10 downto 0);
               dataIn : in STD_LOGIC_VECTOR (15 downto 0);
               clk, wd : in STD_LOGIC;
               dataOut : out STD_LOGIC_VECTOR (15 downto 0));
    end component;
    signal add : STD_LOGIC_VECTOR (10 downto 0);
    signal dataIn : STD_LOGIC_VECTOR (15 downto 0);
    signal clk, wd : STD_LOGIC;
    signal dataOut : STD_LOGIC_VECTOR (15 downto 0);
begin
    md : MemoriaDatos Port map (
        add => add,
        dataIn => dataIn,
        clk => clk,
        wd => wd,
        dataOut => dataOut
    );

    reloj : process begin
        clk <= '0';
        wait for 5 ns;
        clk <= '1';
        wait for 5 ns;
    end process;

    process
        file arch_res : text;   --Apuntadores tipo
        ↪ txt
        variable linea_res : line;
```

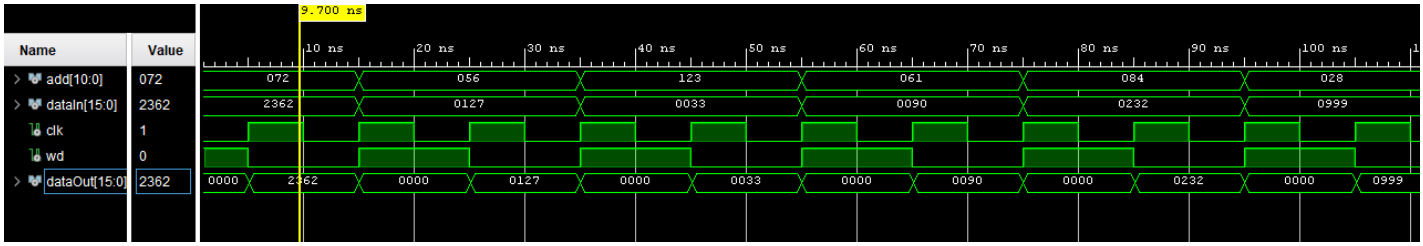
```
42          variable var_data_out : STD_LOGIC_VECTOR (15 downto 0);
43
44          file arch_en : text; --Apuntadores tipo txt
45          variable linea_en: line;
46          variable var_wd : STD_LOGIC;
47          variable var_add : STD_LOGIC_VECTOR (11 downto 0);
48          variable var_data_in : STD_LOGIC_VECTOR (15 downto 0);
49          variable cadena : string (1 to 7);
50      begin
51
52          --- ADD  WD  DATAIN
53          file_open(arch_en, "Estimulos.txt", READ_MODE);
54
55          --- ADD  WD  DATAIN    DATAOUT
56          file_open(arch_res, "Resultado.txt", WRITE_MODE);
57
58          cadena := "    ADD";
59          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪  "ADD"
60          cadena := "     WD";
61          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪  "WD"
62          cadena := " DATAIN";
63          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪  "DATAIN"
64          cadena := "DATAOUT";
65          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪  "DATAOUT"
66
67          writeline(arch_res, linea_res);-- escribe la linea en el archivo
68
69          for i in 0 to 11 loop
70              readline(arch_en, linea_en); -- lee una linea completa
71              --- ADD  WD  DATAIN
72
73              --Lee ADD
74              Hread(linea_en, var_add);
75              add <= var_add(10 downto 0);
76
77              --Lee WD
78              read(linea_en, var_wd);
79              wd <= var_wd;
80
81              --Lee DATAINT
```

```
82              Hread(linea_en, var_data_in);
83              dataIn <= var_data_in;
84
85              wait until rising_edge(clk); --ESPERA AL FLANCO DE SUBIDA
86              var_data_out := dataOut;
87
88              --- ADD  WD  DATAIN    DATAOUT
89              Hwrite(linea_res, var_add, right, 8);--ESCRIBE EL CAMPO ADD
90              write(linea_res, var_wd, right, 8);--ESCRIBE EL CAMPO WD
91              Hwrite(linea_res, var_data_in, right, 8);--ESCRIBE EL CAMPO
                ↪ DATAIN
92              Hwrite(linea_res, var_data_out, right, 8);--ESCRIBE EL CAMPO
                ↪ DATAOUT
93
94              writeline(arch_res, linea_res);-- escribe la linea en el archivo
95          end loop;
96          file_close(arch_en);  -- cierra el archivo
97          file_close(arch_res);  -- cierra el archivo
98          wait;
99      end process;
100
101 end Behavioral;
```

4. Simulación

1. Escritura en la localidad x72 con el valor x2362
2. Lectura de la localidad x72
3. Escritura en la localidad x56 con el valor x127
4. Lectura de la localidad x56
5. Escritura en la localidad x123 con el valor x33
6. Lectura de la localidad x123
7. Escritura en la localidad x61 con el valor x90
8. Lectura de la localidad x61
9. Escritura en la localidad x84 con el valor x232
10. Lectura de la localidad x84
11. Escritura en la localidad x28 con el valor x999
12. Lectura de la localidad x28

4.1. Forma de onda de simulación



4.2. Archivo entrada: Estimulos.txt

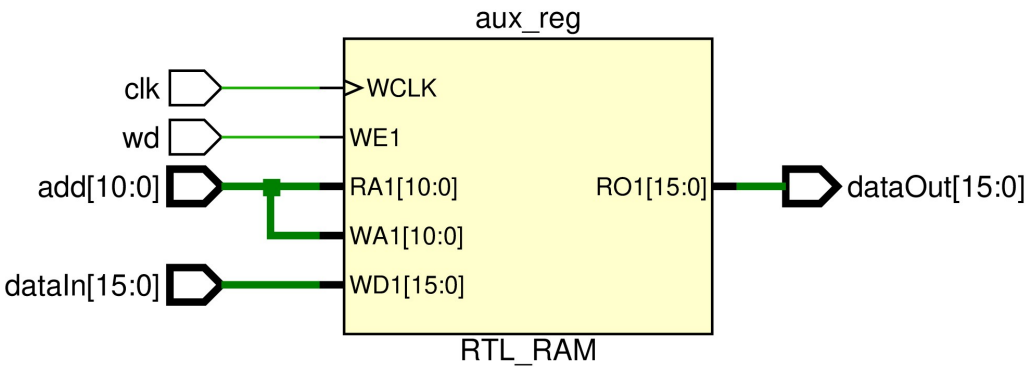
```
--- ADD WD DATAIN
072 1 2362
072 0 2362
056 1 0127
056 0 0127
123 1 0033
123 0 0033
061 1 0090
061 0 0090
084 1 0232
084 0 0232
028 1 0999
028 0 0999
```

4.3. Archivo salida: Resultado.txt

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

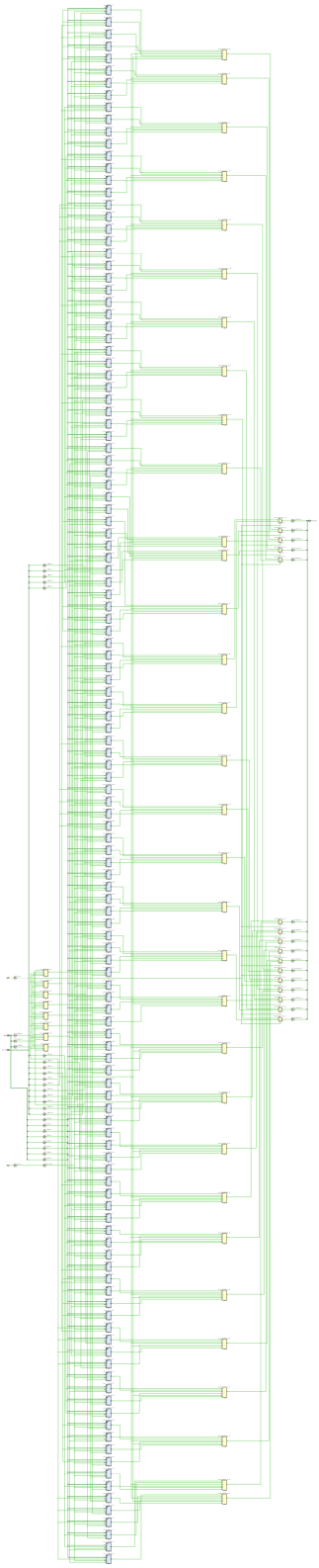
5. Diagramas RTL

5.1. Análisis RTL



5.2. Synthesis

5.2.1. Diagrama comprimido



5.2.2. Diagrama extendido