Reporte de practica 7

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 memoriaDat is
    generic (
        m : integer := 11; --tamanio del bus de direcciones
        n : integer := 16 --tamanio de palabra (dato)
    );
    Port(add : in STD_LOGIC_VECTOR (m-1 downto 0);
         dataIn : in STD_LOGIC_VECTOR (n-1 downto 0);
         wd, clk : in STD_LOGIC;
         dataOut : out STD_LOGIC_VECTOR (n-1 downto 0));

end memoriaDat;

architecture Behavioral of memoriaDat 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;
```

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_arith.ALL;
USE IEEE.STD_LOGIC_unsigned.ALL;


entity tbMemoria is
end tbMemoria;

```

```
13 architecture Behavioral of tbMemoria is
14     component memoriaDat is
15         Port(add : in STD_LOGIC_VECTOR (10 downto 0);
16              dataIn : in STD_LOGIC_VECTOR (15 downto 0);
17              wd, clk : in STD_LOGIC;
18              dataOut : out STD_LOGIC_VECTOR (15 downto 0));
19     end component;
20 
21     signal add:  STD_LOGIC_VECTOR (10 downto 0) :=(others => '0');
22     signal dataIn,dataOut :  STD_LOGIC_VECTOR (15 downto 0) :=(others => '0');
23     signal clk,wd:  STD_LOGIC := '0';
24     signal clk_period : time := 10 ns;
25 begin
26     maping:memoriaDat port map(
27         add=>add,
28         dataIn=>dataIn,
29         wd=>wd,
30         clk=>clk,
31         dataOut=>dataOut
32     );
33     clk_proc:process
34     begin
35         clk<='0';
36         wait for clk_period/2;
37         clk<='1';
38         wait for clk_period/2;
39     end process;
40 
41     test: process
42     --Archivos
43     file arch_res:text;
44     file arch_vec:text;
45 
46     --Salida
47     VARIABLE LINEA_RES : line; --Linea de salida
48     VARIABLE VAR_dataOut : STD_LOGIC_VECTOR(15 DOWNTO 0);
49 
50     --Entrada
51     VARIABLE LINEA_VEC : line; --Vectores de entrada
52    VARIABLE VAR_add : STD_LOGIC_VECTOR(10 DOWNTO 0);
53    VARIABLE VAR_dataIn : STD_LOGIC_VECTOR(15 DOWNTO 0);
54    VARIABLE VAR_WD : STD_LOGIC;
55    VARIABLE CAD : STRING (1 to 8);
56     begin
57     file_open(ARCH_VEC, "/media/d3vcr4ck/externData/materias-Sem20_2/
       arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/memoriaDatos/
       input.txt", READ_MODE);
58     file_open(ARCH_RES, "/media/d3vcr4ck/externData/materias-Sem20_2/
       arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/memoriaDatos/
       output.txt", WRITE_MODE);
59     CAD := "     add";
60     write(LINEA_RES, CAD, right, CAD'LENGTH+1);
61     CAD := "      WD";
62     write(LINEA_RES, CAD, right, CAD'LENGTH+1);
63     CAD := "  dataIn";
64     write(LINEA_RES, CAD, right, CAD'LENGTH+1);
65     CAD := " dataOut";
66     write(LINEA_RES, CAD, right, CAD'LENGTH+1);
67 
68     writeline(ARCH_RES,LINEA_RES);
69 
70     for i in 0 to 11 loop
71         readline(ARCH_VEC,LINEA_VEC); --LECTURA DE LAS LINEAS DEL entrada.txt
72         Hread(LINEA_VEC, VAR_add);
73         add<=VAR_add;
74         read(LINEA_VEC, VAR_WD);
75         WD<=VAR_WD;
```

```
        Hread(LINEA_VEC, VAR_dataIn);
        dataIn<=VAR_dataIn;
        wait until rising_edge(CLK); --Espera hasta el ascenso
        VAR_dataOut := dataOut;
        Hwrite(LINEA_RES, VAR_add, right, 8);
        write (LINEA_RES, VAR_WD, right, 8);
        Hwrite (LINEA_RES, VAR_dataIn, right, 8);
        Hwrite (LINEA_RES, VAR_dataOut, right, 8);
        writeline(ARCH_RES,LINEA_RES);
    end loop;
    file_close(ARCH_VEC);
   file_close(ARCH_RES);
   wait;
    end process;
end Behavioral;
```

Archivo de entrada (input.txt)

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

--add WD dataIn
```

Archivo de salida (output.txt)

```
      add        WD    dataIn   dataOut
     072         1     2362     XXXX
     072         0     0000     2362
     056         1     0127     XXXX
     056         0     0000     0127
     123         1     0033     XXXX
     123         0     0000     0033
     061         1     0090     XXXX
     061         0     0000     0090
     084         1     0232     XXXX
     084         0     0000     0232
     028         1     0999     XXXX
     028         0     0000     0999
```

3. Simulaciones

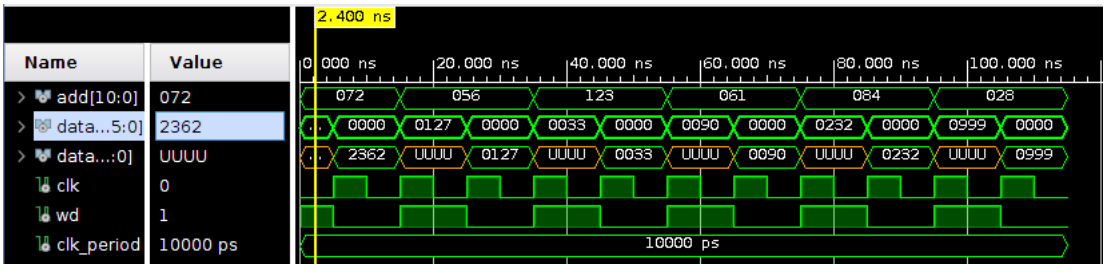

Figura general de la forma de onda del Test-Bench

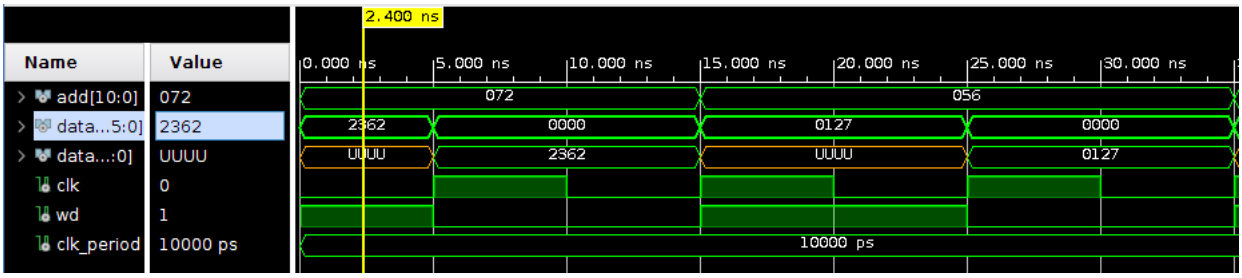

Figura 1

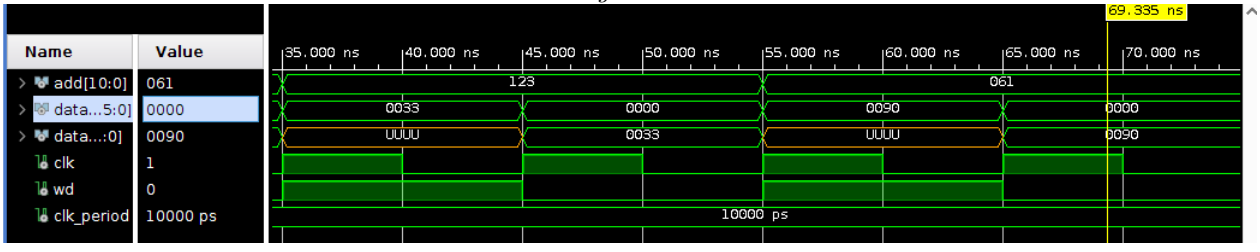

Figura 2

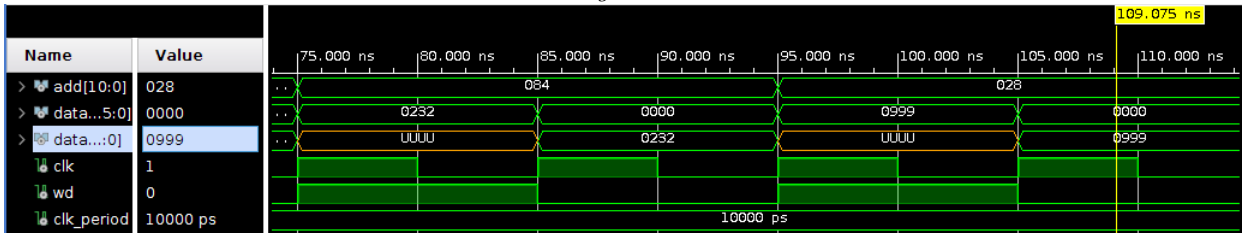

Figura 3

4. Diagrama RTL