Martínez Coronel Brayan Yosafat

CÓDIGO VHDL

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

entity memoria is
    Port ( readReg1, readReg2, writeReg : in STD_LOGIC_VECTOR (3 downto 0);
           shamt : in STD_LOGIC_VECTOR (3 downto 0);
           writeData : in STD_LOGIC_VECTOR (15 downto 0);
           clk, clear, dir, SHE, WR : in STD_LOGIC;
           readData1, readData2 : out STD_LOGIC_VECTOR(15 downto 0)
           );
end memoria;

architecture Behavioral of memoria is
type arreglo is array (0 to 15) of STD_LOGIC_VECTOR (15 downto 0);
signal mem : arreglo;
begin
    process (clk, clear)
    begin
        if (clear = '1') then
            for i in 0 to 15 loop
                mem(i) <= "0000000000000000";
            end loop;
        elsif (rising_edge(clk)) then
            if (WR = '1') then
                if (SHE = '1') then
                    if (dir = '1') then
                        mem(conv_integer(writeReg)) <= to_stdlogicvector(to_bitvector(mem(conv_integer(readReg1)))
sll (conv_integer(shamt)));
                    else
                        mem(conv_integer(writeReg)) <= to_stdlogicvector(to_bitvector(mem(conv_integer(readReg1)))
srl (conv_integer(shamt)));
                    end if;
                else
                    mem(conv_integer(writeReg)) <= writeData;
                end if;
            else
                mem <= mem;
            end if;
        end if;
    end process;
    readData1 <= mem(conv_integer(readReg1));
    readData2 <= mem(conv_integer(readReg2));

end Behavioral;
```

CÓDIGO DEL TEST-BENCH

```
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
component memoria is Port ( readReg1, readReg2, writeReg : in STD_LOGIC_VECTOR (3 downto 0);
           shamt : in STD_LOGIC_VECTOR (3 downto 0);
           writeData : in STD_LOGIC_VECTOR (15 downto 0);
           clk, clear, dir, SHE, WR : in STD_LOGIC;
           readData1, readData2 : out STD_LOGIC_VECTOR(15 downto 0)
           );
end component;
signal readReg1, readReg2, writeReg : STD_LOGIC_VECTOR (3 downto 0);
signal shamt : STD_LOGIC_VECTOR (3 downto 0);
signal writeData : STD_LOGIC_VECTOR (15 downto 0);
signal clk, clear, dir, SHE, WR : STD_LOGIC;
signal readData1, readData2 : STD_LOGIC_VECTOR(15 downto 0);
constant clk_period : time := 10 ns;
begin
    test: memoria port map (
        readReg1 =>  readReg1,
        readReg2 => readReg2,
        writeReg => writeReg,
        shamt => shamt,
        writeData => writeData,
        clk => clk,
        clear => clear,
        dir => dir,
        SHE => SHE,
        WR => WR,
        readData1 => readData1,
        readData2 => readData2
    );
```

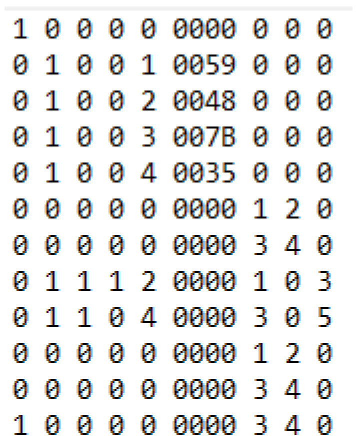
```
  clk_process : process
  begin
                    CLK <= '0';
                      wait for clk_period/2;
                    CLK <= '1';
                      wait for clk_period/2;
  end process;

   stim_proc: process
   file ARCH_SALIDA : TEXT;

   variable LINEA_SALIDA : line;
   variable var_result_1 : STD_LOGIC_VECTOR(15 DOWNTO 0);
   variable var_result_2 : STD_LOGIC_VECTOR(15 DOWNTO 0);

   file ARCH_ENTRADA : TEXT;
   variable LINEA_ENTRADA : line;
   variable v_readReg1, v_readReg2, v_writeReg : STD_LOGIC_VECTOR (3 downto 0);
   variable v_shamt : STD_LOGIC_VECTOR (3 downto 0);
   variable v_writeData : STD_LOGIC_VECTOR (15 downto 0);
   variable v_clear, v_dir, v_SHE, v_WR : STD_LOGIC;
   variable CADENA : STRING(1 TO 6);
   begin
      file_open(ARCH_ENTRADA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica5\entradas.txt",
READ_MODE);
      file_open(ARCH_SALIDA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica5\salidas.txt",
WRITE_MODE);

      CADENA := "   RR1";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "   RR2";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := " SHAMT";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "  WREG";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "    WD";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "    WR";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "   SHE";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "   DIR";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "   RD1";
      write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);
      CADENA := "   RD2";
```

```
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

   read(LINEA_ENTRADA, v_clear);
   clear <= v_clear;
   read(LINEA_ENTRADA, v_WR);
   WR <= v_WR;
   read(LINEA_ENTRADA, v_SHE);
   SHE <= v_SHE;
   read(LINEA_ENTRADA, v_dir);
   dir <= v_dir;

   Hread(LINEA_ENTRADA, v_writeReg);
   writeReg <= v_writeReg;
   Hread(LINEA_ENTRADA, v_writeData);
   writeData <= v_writeData;
   Hread(LINEA_ENTRADA, v_readReg1);
   readReg1 <= v_readReg1;
   Hread(LINEA_ENTRADA, v_readReg2);
   readReg2 <= v_readReg2;
   Hread(LINEA_ENTRADA, v_shamt);
   shamt <= v_shamt;

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

   var_result_1 := readData1;
   var_result_2 := readData2;

   Hwrite(LINEA_SALIDA, v_readReg1, right, 6);
   Hwrite(LINEA_SALIDA, v_readReg2, right, 7);
   write(LINEA_SALIDA, v_shamt, right, 8);
   Hwrite(LINEA_SALIDA, v_writeReg, right, 6);
   Hwrite(LINEA_SALIDA, v_writeData, right, 9);
   write(LINEA_SALIDA, v_WR, right, 6);
   write(LINEA_SALIDA, v_SHE, right, 6);
   write(LINEA_SALIDA, v_dir, right, 7);
   Hwrite(LINEA_SALIDA, readData1, right, 8);
   Hwrite(LINEA_SALIDA, readData2, right, 7);

   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;
```

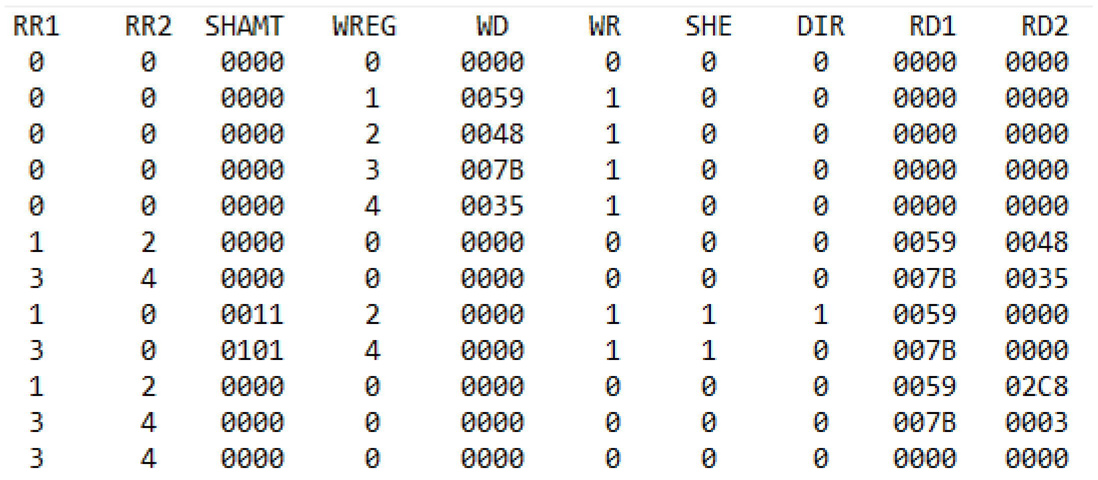
ARCHIVOS

ENTRADA



```
1 0 0 0 0 0000 0 0 0
0 1 0 0 1 0059 0 0 0
0 1 0 0 2 0048 0 0 0
0 1 0 0 3 007B 0 0 0
0 1 0 0 4 0035 0 0 0
0 0 0 0 0 0000 1 2 0
0 0 0 0 0 0000 3 4 0
0 1 1 1 2 0000 1 0 3
0 1 1 0 4 0000 3 0 5
0 0 0 0 0 0000 1 2 0
0 0 0 0 0 0000 3 4 0
1 0 0 0 0 0000 3 4 0
```

SALIDA



```
RR1    RR2   SHAMT   WREG    WD     WR    SHE    DIR    RD1    RD2
 0      0    0000     0     0000    0      0      0    0000   0000
 0      0    0000     1     0059    1      0      0    0000   0000
 0      0    0000     2     0048    1      0      0    0000   0000
 0      0    0000     3     007B    1      0      0    0000   0000
 0      0    0000     4     0035    1      0      0    0000   0000
 1      2    0000     0     0000    0      0      0    0059   0048
 3      4    0000     0     0000    0      0      0    007B   0035
 1      0    0011     2     0000    1      1      1    0059   0000
 3      0    0101     4     0000    1      1      0    007B   0000
 1      2    0000     0     0000    0      0      0    0059   02C8
 3      4    0000     0     0000    0      0      0    007B   0003
 3      4    0000     0     0000    0      0      0    0000   0000
```

DIAGRAMAS RTL