INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

U.A: Arquitectura de Computadoras

“Práctica 7: Archivo de Registros”

Grupo: 3CV11

Profesor: Vega García Nayeli

Sánchez Becerra Ernesto Daniel

Código de implementación

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity FileReg is
    Port ( writeReg, readReg1,readReg2,shamt : in STD_LOGIC_VECTOR (3 downto 0);
           writeData : in STD_LOGIC_VECTOR (15 downto 0);
           clk,clr,wr,she,dir : in STD_LOGIC;
           readData1,readData2 : out STD_LOGIC_VECTOR (15 downto 0));
end FileReg;

architecture Behavioral of FileReg is
type palabra is array (0 to 15) of std_logic_vector(writeData'length-1 downto 0);
signal banco: palabra;
begin
    process(clk,clr)
    variable posBanco,readl,exponente: integer;
    variable auxShift,resShift: std_logic_vector(writeData'length-1 downto 0);
    variable  N : integer := writeData'length;
    begin
        readl := conv_integer(readReg1);
        posBanco :=conv_integer(writeReg);
        if(clr='1') then
            banco<=(others=>(others=>'0'));
        elsif(rising_edge(clk)) then
            if(wr='1' and she='0') then
                banco(posBanco) <= writeData;
        elsif(wr='1' and she='1' and dir='0') then -- shift right
            auxShift := banco(readl);
            resShift := banco(readl);
            for i in 0 to shamt'length-1 loop
                exponente:= 2**i;
                for j in 0 to N-exponente-1 loop
                    if(shamt(i)='1') then
                        resShift(j) := auxShift(j + exponente);
                    end if;
                end loop;
                for j in N-exponente to N-1 loop
                    if(shamt(i)='1') then
                        resShift(j) := '0';
                    end if;
                end loop;
                auxShift:=resShift;
            end loop;
            banco(posBanco) <= resShift;
        elsif(wr='1' and she='1' and dir='1') then -- shift left
            auxShift := banco(readl);
            resShift := banco(readl);
            for i in 0 to shamt'length-1 loop
                exponente:= 2**i;
                for j in 0 to exponente-1 loop
                    if(shamt(i)='1') then
                        resShift(j) := '0';
                    end if;
                end loop;

                for j in exponente to N-1 loop
                    if(shamt(i)='1') then
                        resShift(j) := auxShift(j - exponente);
                    end if;
                end loop;
                auxShift:=resShift;
            end loop;
            banco(posBanco) <= resShift;
        end if;
    end if;
    end process;
    readData1 <=banco(conv_integer(readReg1));
    readData2 <=banco(conv_integer(readReg2));


end Behavioral;
```

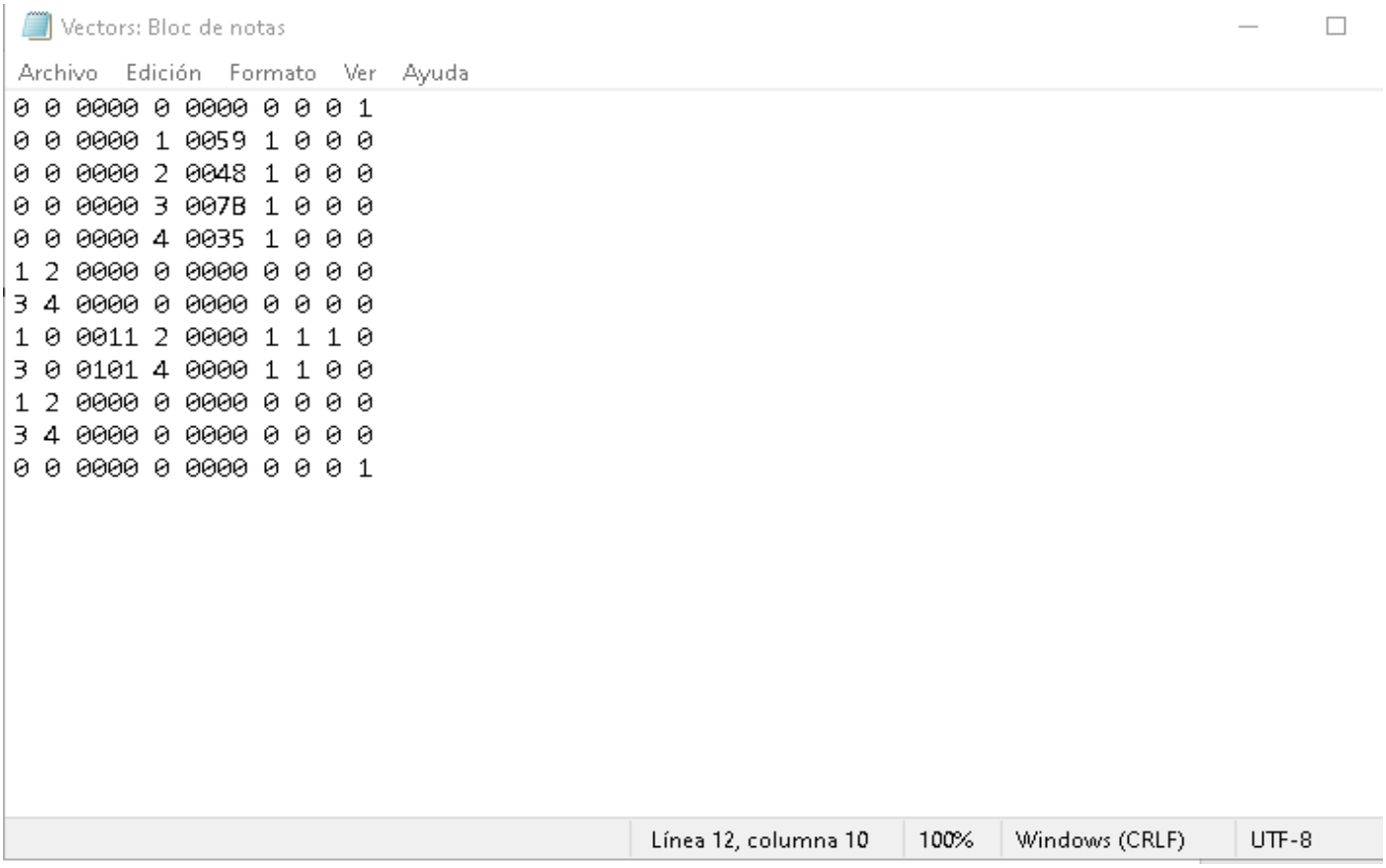
Código de simulación

```
library IEEE;
library std;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_textio.all;
use ieee.std_logic_arith.all;
use std.textio.all;

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

architecture Behavioral of FileReg_TB is

    component FileReg
        Port (
            writeReg : in STD_LOGIC_VECTOR (3 downto 0);
            readReg1 : in STD_LOGIC_VECTOR (3 downto 0);
            readReg2 : in STD_LOGIC_VECTOR (3 downto 0);
            shamt : in STD_LOGIC_VECTOR (3 downto 0);
            writeData : in STD_LOGIC_VECTOR (15 downto 0);
            clk: in STD_LOGIC;
            clr : in STD_LOGIC;
            wr : in STD_LOGIC;
            she : in STD_LOGIC;
            dir : in STD_LOGIC;
            readData1 : out STD_LOGIC_VECTOR (15 downto 0);
            readData2 : out STD_LOGIC_VECTOR (15 downto 0)
          );
    end component;
    --entradas
            signal writeReg : STD_LOGIC_VECTOR (3 downto 0) := ( others => '0' );
            signal readReg1 : STD_LOGIC_VECTOR (3 downto 0) := ( others => '0' );
            signal readReg2 : STD_LOGIC_VECTOR (3 downto 0) := ( others => '0' );
            signal shamt : STD_LOGIC_VECTOR (3 downto 0) := ( others => '0' );
            signal writeData : STD_LOGIC_VECTOR (15 downto 0) := ( others => '0' );
            signal clk: STD_LOGIC := '0';
            signal clr : STD_LOGIC := '0';
            signal wr : STD_LOGIC := '0';
            signal she : STD_LOGIC := '0';
            signal dir : STD_LOGIC := '0';
    --Salidas
            signal readData1 : STD_LOGIC_VECTOR (15 downto 0);
            signal readData2 : STD_LOGIC_VECTOR (15 downto 0);

    --Definición del reloj
            constant clock_period : time := 10 ns;
begin
    uut: FileReg port map(
            writeReg => writeReg,
            readReg1 => readReg1,
            readReg2 => readReg2,
            shamt => shamt,
            writeData => writeData,
            clk => clk, --clock
            clr => clr, --clear
            wr => wr,
            she => she,
            dir => dir,
            readData1 => readData1,
            readData2 => readData2
          );
```

Código de simulación

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

    stimulus: process
        file file_vectors : text;
        variable line_vector : line;
        variable var_rr1 : std_logic_vector ( 3 downto 0 ); -- Read Register 1.
        variable var_rr2 : std_logic_vector ( 3 downto 0 ); -- Read Register 2.
        variable var_shamt : std_logic_vector ( 3 downto 0 ); -- Shamt.
        variable var_wreg : std_logic_vector ( 3 downto 0 ); -- Write Register.
        variable var_wd : std_logic_vector ( 15 downto 0 ); -- Write data.
        variable var_wr : std_logic;
        variable var_she : std_logic;
        variable var_dir : std_logic;
        variable var_clear: std_logic;

        file file_results : text;
        variable line_result : line;
        variable var_rd1 : std_logic_vector ( 15 downto 0 );
        variable var_rd2 : std_logic_vector ( 15 downto 0 );
        variable str : string ( 1 to 7 );

        begin
            file_open ( file_vectors, "Vectors.txt", read_mode );
            file_open ( file_results, "Results.txt", write_mode );

            str := "    RR1";
            write ( line_result, str, right, str'length + 1 );
            str := "    RR2";
            write ( line_result, str, right, str'length + 1 );
            str := "  SHAMT";
            write ( line_result, str, right, str'length + 1 );
            str := "   WREG";
            write ( line_result, str, right, str'length + 1 );
            str := "  WDATA";
            write ( line_result, str, right, str'length + 1 );
            str := "     WR";
            write ( line_result, str, right, str'length + 1 );
            str := "    SHE";
            write ( line_result, str, right, str'length + 1 );
            str := "    DIR";
            write ( line_result, str, right, str'length + 1 );
            str := "    RD1";
            write ( line_result, str, right, str'length + 1 );
            str := "    RD2";
            write ( line_result, str, right, str'length + 1 );
            writeline ( file_results, line_result );

            wait for 100 ns;

            for i in 0 to 11 loop
                readline ( file_vectors, line_vector );
                Hread ( line_vector, var_rr1 );
                readReg1 <= var_rr1;
                Hread ( line_vector, var_rr2 );
                readReg2 <= var_rr2;
                read ( line_vector, var_shamt );
                shamt <= var_shamt;
                Hread ( line_vector, var_wreg );
                writeReg <= var_wreg;
                Hread ( line_vector, var_wd );
                writeData <= var_wd;
                read ( line_vector, var_wr );
                wr <= var_wr;
                read ( line_vector, var_she );
                she <= var_she;
                read ( line_vector, var_dir );
                dir <= var_dir;
                read ( line_vector, var_clear );
                clr <= var_clear;

                wait until rising_edge ( clk );

                var_rd1 := readData1;
                var_rd2 := readData2;
                Hwrite ( line_result, var_rr1, right, 8 );
                Hwrite ( line_result, var_rr2, right, 8 );
                Hwrite ( line_result, var_shamt, right, 8 );
                Hwrite ( line_result, var_wreg, right, 8 );
                Hwrite ( line_result, var_wd, right, 8 );
                write ( line_result, var_wr, right, 8 );
                write ( line_result, var_she, right, 8 );
                write ( line_result, var_dir, right, 8 );
                Hwrite ( line_result, var_rd1, right, 8 );
                Hwrite ( line_result, var_rd2, right, 8 );

                writeline ( file_results, line_result );
            end loop;

            file_close ( file_vectors );
            file_close ( file_results );

            wait;
    end process;
end;
```

Archivo con vectores de entrada



Archivo con tabla de resultados

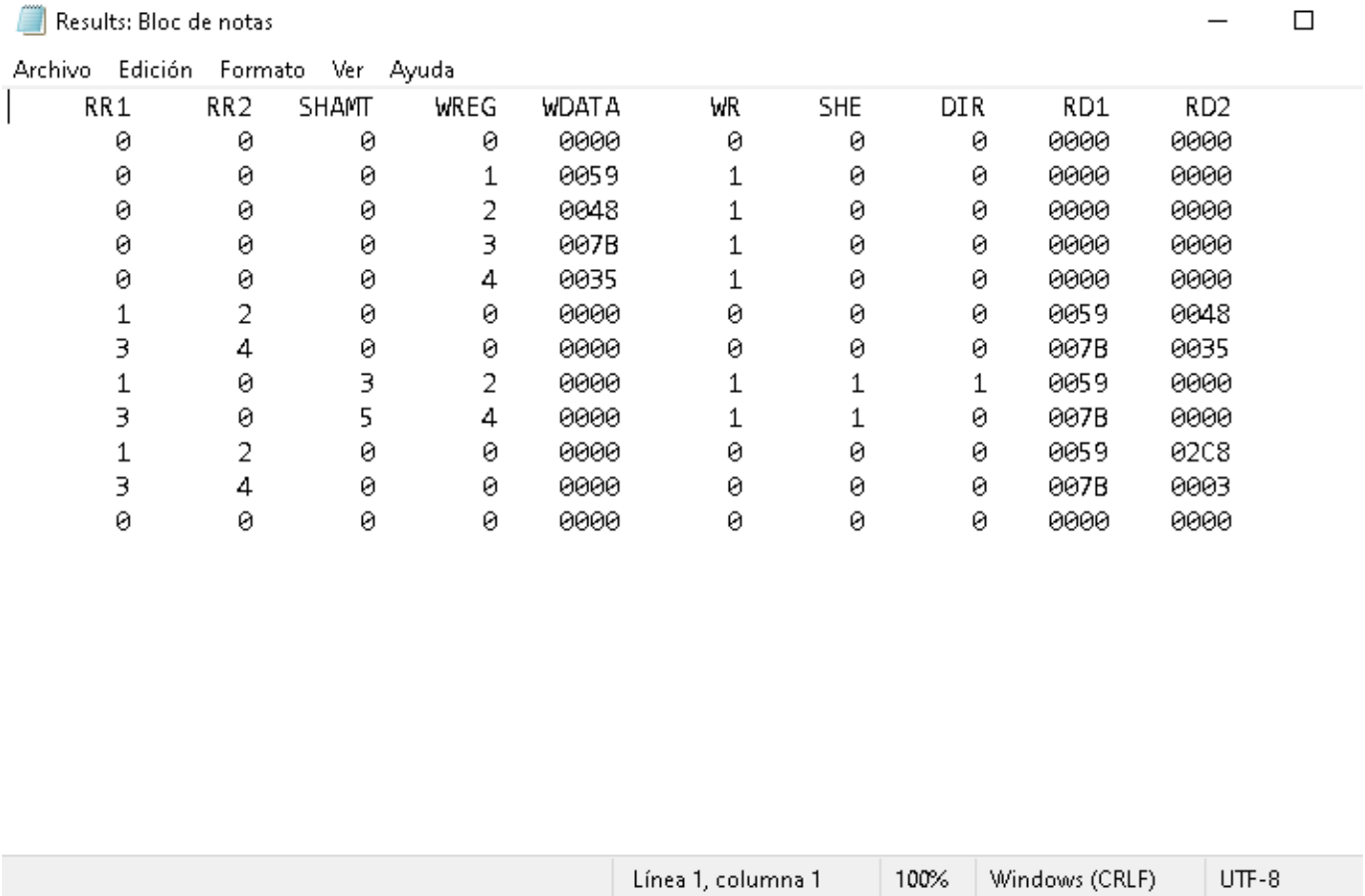


Diagrama RTL