Código de implementación:

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

entity registro is
   Port (
      writeData: in std_logic_vector (15 downto 0);
      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 );
      WR: in std_logic;
      DIR: in std_logic;
      SHE: in std_logic;
      clr: in std_logic;
      clk: in std_logic;
      readData1: out std_logic_vector ( 15 downto 0 );
      readData2: out std_logic_vector ( 15 downto 0 )
   );
end registro;

architecture Behavioral of registro is

type arreglo is array (0 to 15) of std_logic_vector( 15 downto 0 );
signal registros: arreglo;

begin
   -- Lectura del archivo de registros
   readData1 <= registros( conv_integer( readReg1 ) );
   readData2 <= registros( conv_integer( readReg2 ) );
   -- Carga en el archivo de registros
   process ( clk, clr )
   variable shift: bit_vector ( 15 downto 0 );
   variable auxq: std_logic_vector ( 15 downto 0 );
   begin
      shift := to_bitvector( registros ( conv_integer( readReg2 ) ) );
      if ( clr = '1' )then
         registros <= (others => (others => '0'));
      elsif ( rising_edge(clk) ) then -- Señales síncronas
         if ( WR = '1' and SHE = '0') then -- Carga
            registros(conv_integer( writeReg )) <= writeData;
         elsif ( WR = '1' and SHE = '1' ) then -- Corrimiento
            if ( DIR = '0' ) then -- Corrimiento derecha
               shift := shift srl conv_integer( shamt );
            else -- Corimiento izquierda
```

```
                shift := shift sll conv_integer( shamt );
            end if;
            registros(conv_integer( writeReg )) <= to_stdlogicvector( shift );
        end if;
    end if;
  end process;
end Behavioral;
```

Código de simulación:

```
LIBRARY IEEE;
LIBRARY STD;
USE STD.TEXTIO.ALL;
USE ieee.std_logic_TEXTIO.ALL;

USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_UNSIGNED.ALL;
USE ieee.std_logic_ARITH.ALL;
entity fileReg_TB is
--  Port ( );
end fileReg_TB;

architecture Behavioral of fileReg_TB is

component registro is
    Port (
        writeData: in std_logic_vector (15 downto 0);
        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 );
        WR: in std_logic;
        DIR: in std_logic;
        SHE: in std_logic;
        clr: in std_logic;
        clk: in std_logic;
        readData1: out std_logic_vector ( 15 downto 0 );
        readData2: out std_logic_vector ( 15 downto 0 )
    );
end component;
signal writeData: std_logic_vector (15 downto 0):=(others=>'0');
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 WR:  std_logic:='0';
```

```
signal DIR:  std_logic:='0';
signal SHE:  std_logic:='0';
signal clk:  std_logic:='0';
signal clr:  std_logic:='0';
signal readData1: std_logic_vector ( 15 downto 0 );
signal readData2: std_logic_vector ( 15 downto 0 );
begin

asignacion :  registro
   Port map(
      writeData => writeData,
      writeReg => writeReg,
      readReg1 => readReg1,
      readReg2 => readReg2,
      shamt => shamt,
      WR => WR,
      DIR => DIR,
      SHE => SHE,
      clk => clk,
      clr => clr,
      readData1 => readData1,
      readData2 => readData2
   );

    process
      begin
         clk <= '0';
         wait for 15 ns;
         clk <= '1';
         wait for 15 ns;
        end process;

   stim_proc: process
         file ARCH_RES : TEXT;


         variable LINEA_RES : line;
         -- Variables para las entradas
         variable var_writeReg, var_shamt : STD_LOGIC_VECTOR (3 downto 0);
   variable var_writeData : STD_LOGIC_VECTOR (15 downto 0);
   variable var_readReg1 : STD_LOGIC_VECTOR (3 downto 0);
   variable var_readReg2 : STD_LOGIC_VECTOR (3 downto 0);
   variable var_WR , var_dir, var_SHE, var_clr : STD_LOGIC;
file ARCH_VEC : TEXT;
         variable LINEA_VEC : line;
         VARIABLE CADENA : STRING(1 TO 7);
         -- Variables para los puertos de salida
         variable var_readData1 : STD_LOGIC_VECTOR (15 downto 0);
```

```
        variable var_readData2 : STD_LOGIC_VECTOR (15 downto 0);

  begin
                file_open(ARCH_VEC,
"C:\Users\sebas\Desktop\PracticasArq\Practica5\vectores.txt", READ_MODE);
                file_open(ARCH_RES,
"C:\Users\sebas\Desktop\PracticasArq\Practica5\resultado.txt", WRITE_MODE);


                -- write (   linea  , valor a escribir, lado a escribir, tamaño)
                CADENA := "RR1    ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
      CADENA := "RR2    ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "SHAMT  ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "WREG   ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "WD     ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "WR     ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "SHE    ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "DIR    ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "RD1    ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "RD2    ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                writeline(ARCH_RES,LINEA_RES);-- escribe la linea en el archivo

                WAIT FOR 100 NS;
                FOR I IN 0 TO 11 LOOP
                        readline(ARCH_VEC,LINEA_VEC); -- lee una linea completa
        read(LINEA_VEC, var_clr);
        clr <= var_clr;
        read(LINEA_VEC, var_WR);
        WR <= var_WR;
        read(LINEA_VEC, var_SHE);
        SHE <= var_SHE;
        read(LINEA_VEC, var_dir);
        dir <= var_dir;
        hread(LINEA_VEC, var_shamt);
        shamt <= var_shamt;
        hread(LINEA_VEC, var_writeReg);
        writeReg <= var_writeReg;
        read(LINEA_VEC, var_writeData);
```

```
        writeData <= var_writeData;
        hread(LINEA_VEC, var_readReg1);
        readReg1 <= var_readReg1;
        hread(LINEA_VEC, var_readReg2);
        readReg2 <= var_readReg2;

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

        var_readData1 := readData1;
        var_readData2 := readData2;

        hwrite(LINEA_RES, var_readReg1, left, 8);
        hwrite(LINEA_RES, var_readReg2, left, 8);
        hwrite(LINEA_RES, var_shamt, left, 8);
        hwrite(LINEA_RES, var_writeReg, left, 8);
        hwrite(LINEA_RES, var_writeData, left, 8);
        write(LINEA_RES, var_WR, left, 8);
        write(LINEA_RES, var_SHE, left, 8);
        write(LINEA_RES, var_dir, left, 8);
        hwrite(LINEA_RES, var_readData1, left, 8);
        hwrite(LINEA_RES, var_readData2, left, 18);
                        writeline(ARCH_RES,LINEA_RES);-- escribe la linea en el archivo
                end loop;
                file_close(ARCH_VEC);  -- cierra el archivo
                file_close(ARCH_RES);  -- cierra el archivo

    wait;
  end process;

end Behavioral;
```

Vectores:

vectores: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
1 0 0 0 0 0 0000000000000000 0 0
0 1 0 0 0 1 0000000001011001 0 0
0 1 0 0 0 2 0000000001001000 0 0
0 1 0 0 0 3 0000000001111011 0 0
0 1 0 0 0 4 0000000000110101 0 0
0 0 0 0 0 0 0000000000000000 1 2
0 0 0 0 0 0 0000000000000000 3 4
0 1 1 1 3 2 0000000000000000 0 1
0 1 1 0 5 4 0000000000000000 0 3
0 0 0 0 0 0 0000000000000000 1 2
0 0 0 0 0 0 0000000000000000 3 4
1 0 0 0 0 0 0000000000000000 0 0
```

100% Windows (CRLF) UTF-8

Resultados:

resultado: Bloc de notas

Archivo Edición Formato Ver Ayuda

RR1	RR2	SHAMT	WREG	WD	WR	SHE	DIR	RD1	RD2
0	0	0	0	0000	0	0	0	0000	0000
0	0	0	1	0059	1	0	0	0000	0000
0	0	0	2	0048	1	0	0	0000	0000
0	0	0	3	007B	1	0	0	0000	0000
0	0	0	4	0035	1	0	0	0000	0000
1	2	0	0	0000	0	0	0	0059	0048
3	4	0	0	0000	0	0	0	007B	0035
0	1	3	2	0000	1	1	1	0000	0059
0	3	5	4	0000	1	1	0	0000	007B
1	2	0	0	0000	0	0	0	0059	02C8
3	4	0	0	0000	0	0	0	007B	0003
0	0	0	0	0000	0	0	0	0000	0000

Línea 1, columna 1 100% Windows (CRLF) UTF-8

Diagrama RTL: