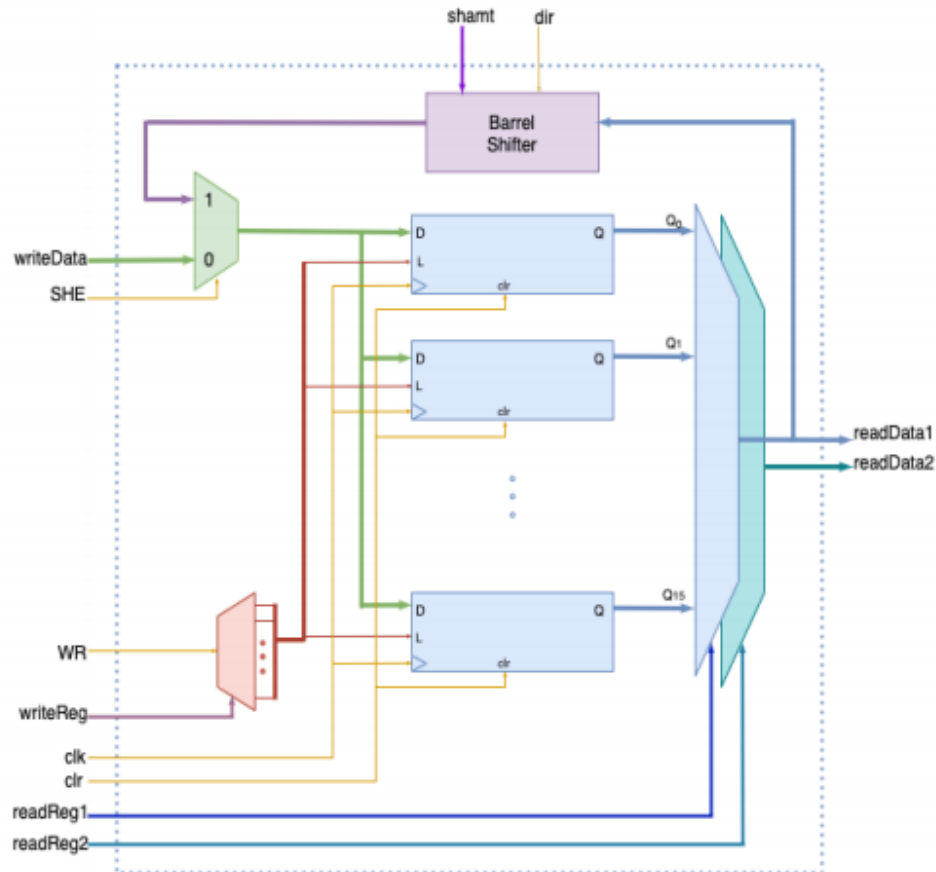


Práctica 6

1.

1. Implemente la arquitectura del archivo de registros, que se muestra en la siguiente figura, de manera modular. Su código debe contener lo siguiente:
 - a. Instanciación de componentes
 - b. Uso de arreglos
 - c. Sentencias de control para ciclos

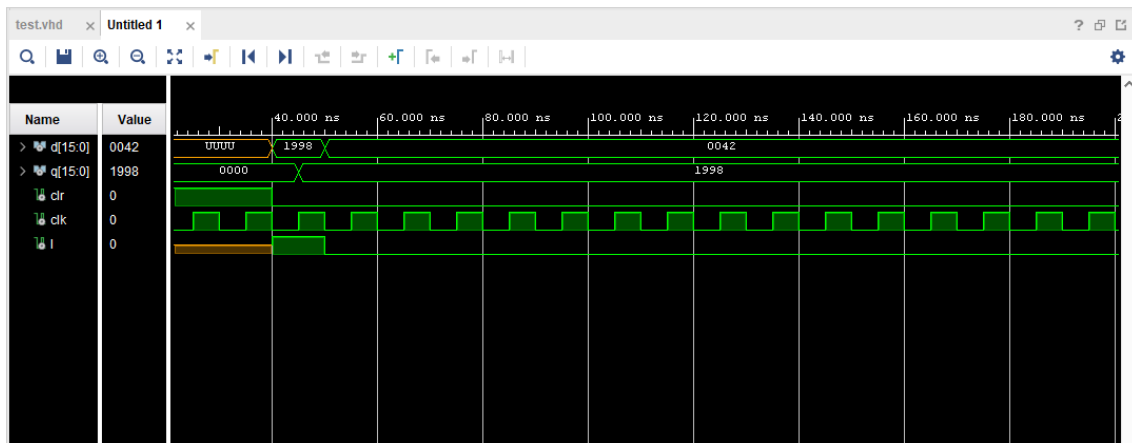


2.

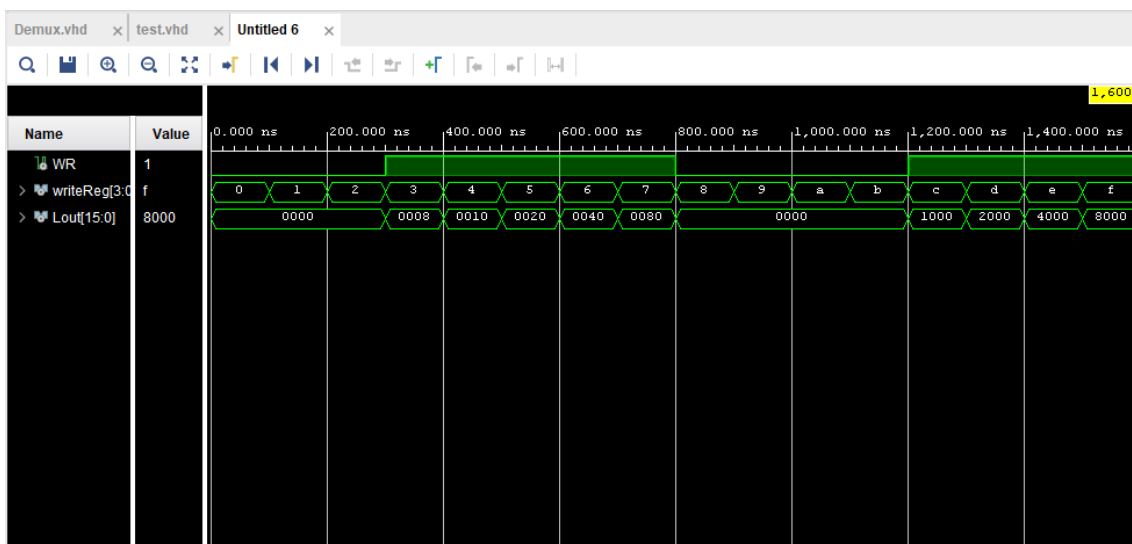
2. Los elementos a ocupar se describen a continuación:
 - a. Registro: Elemento de memoria que permite el reset asíncrono y la carga síncrona.
 - b. Multiplexor de 16 canales: Permite la selección de uno de los canales mediante el vector selector de entrada.
 - c. Demultiplexor de 16 canales de salida: Permite la selección del registro en el cual se realizará la carga de información.
 - d. Barrel-shifter: Elemento combinatorio que permite realizar corrimientos a la derecha o a la izquierda, según se seleccione.

3.

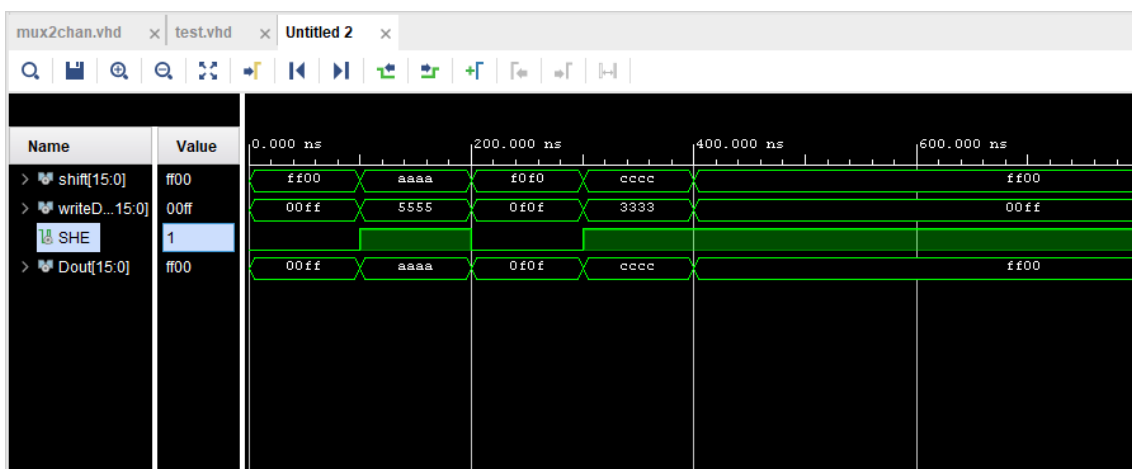
Simulación Registro



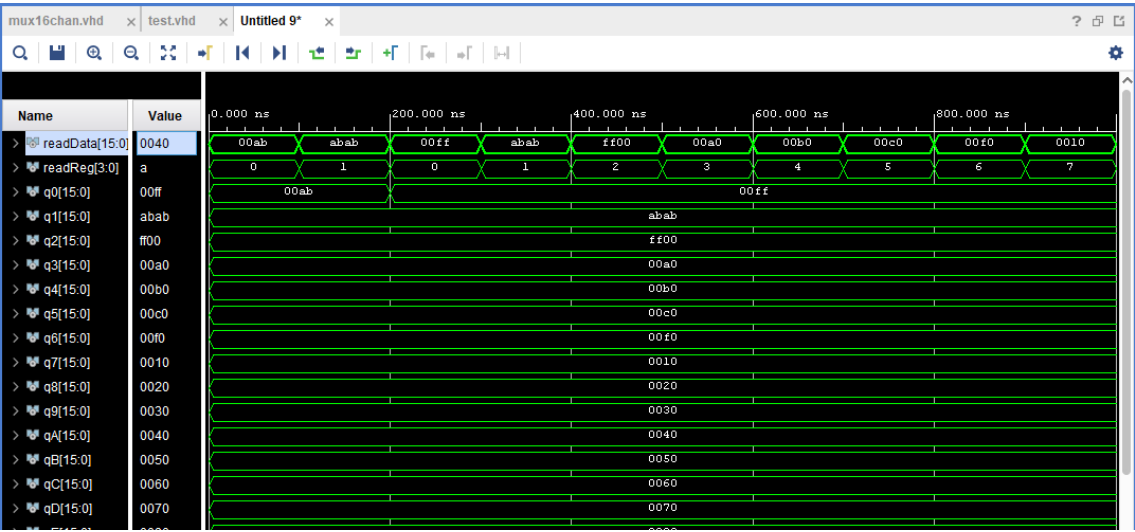
Simulación Demux



Simulación MUX 2 canales



Simulación MUX 16 canales



Simulación BarrelShifter

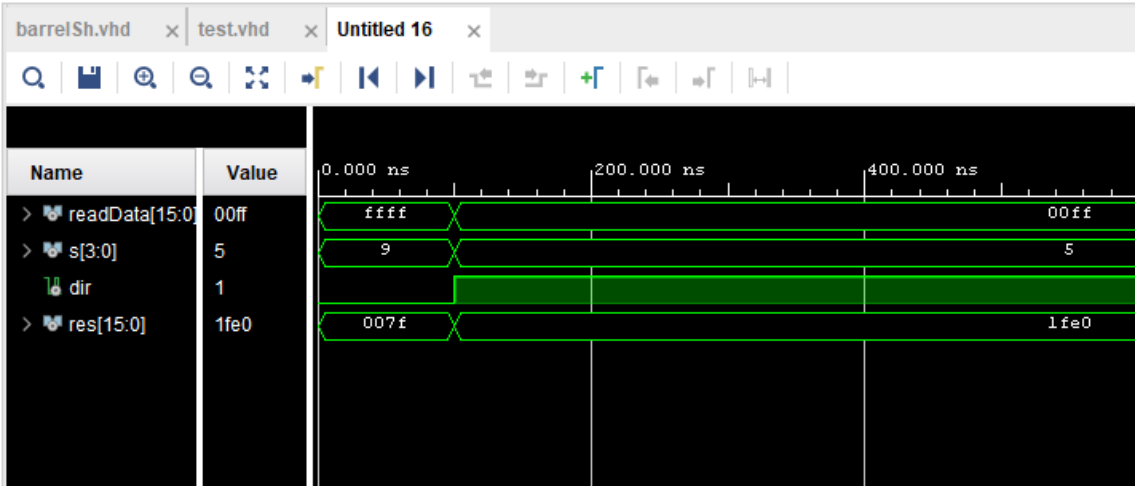


Diagrama RTL Registro

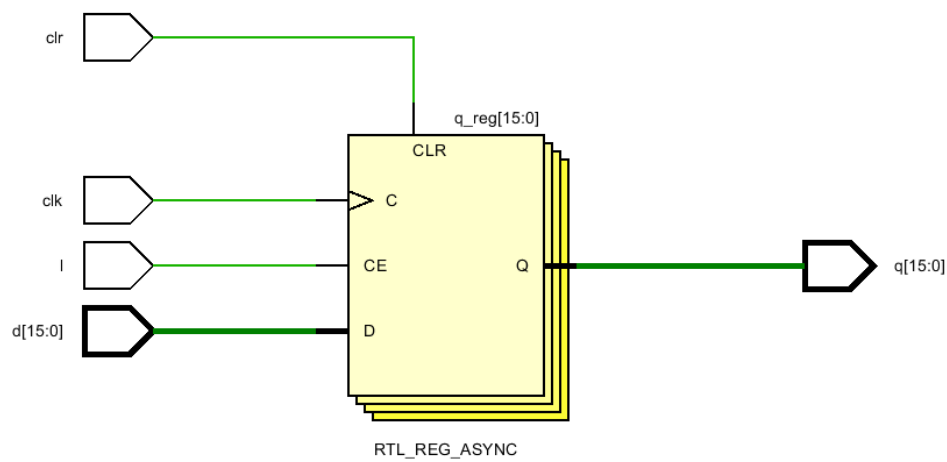


Diagrama RTL DEMUX

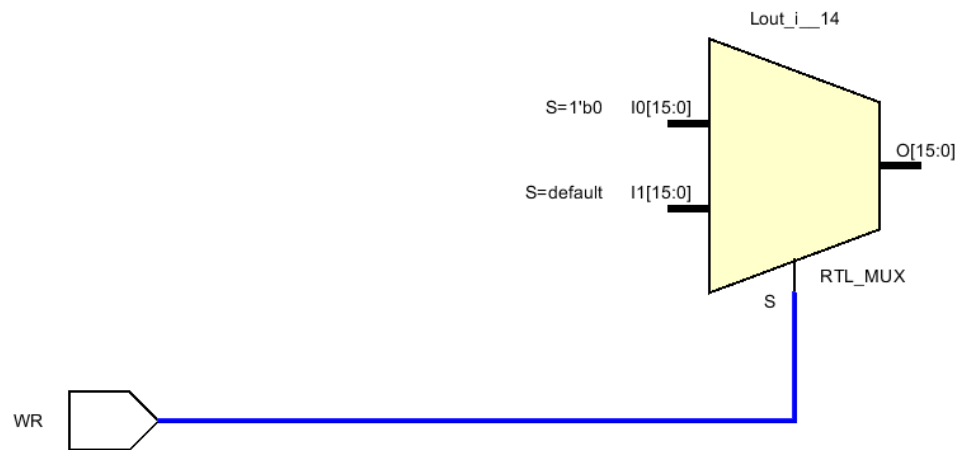


Diagrama RTL Multiplexor 2 Canales

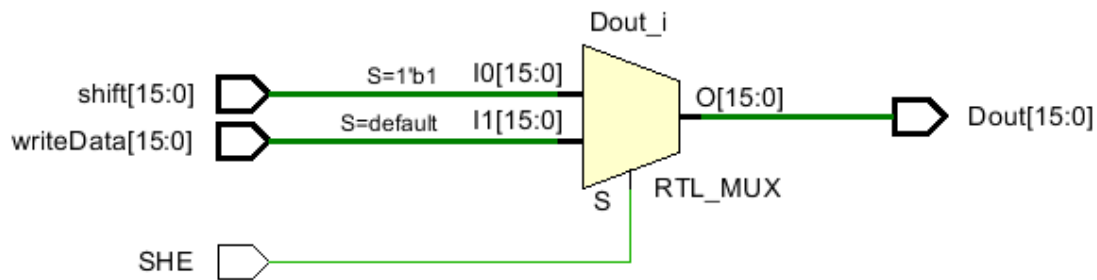


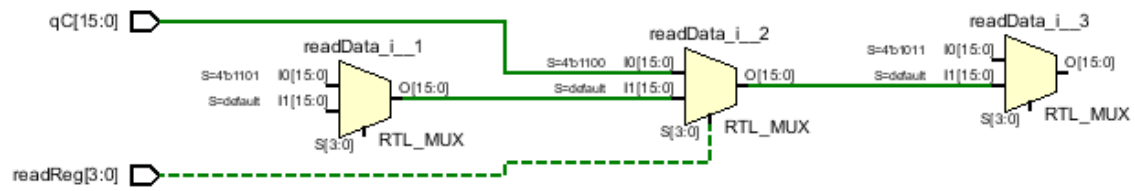
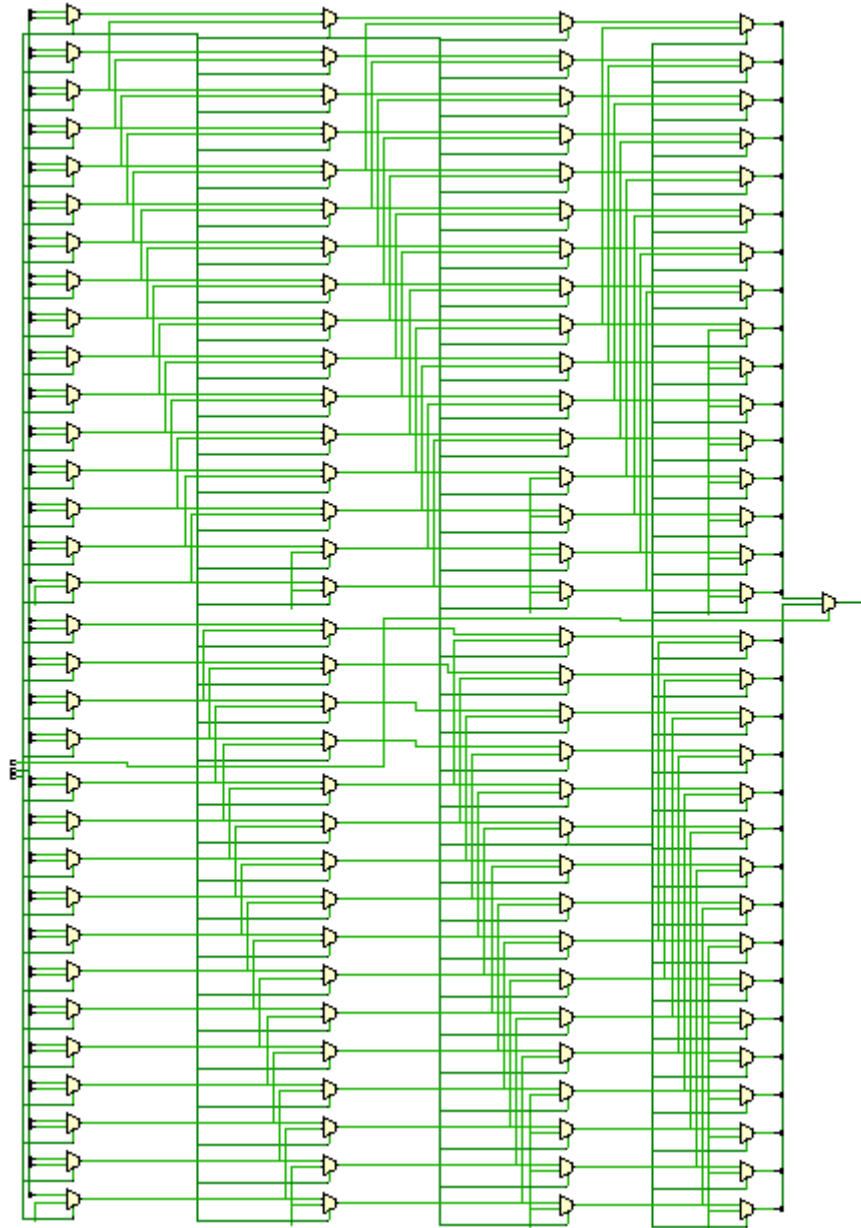
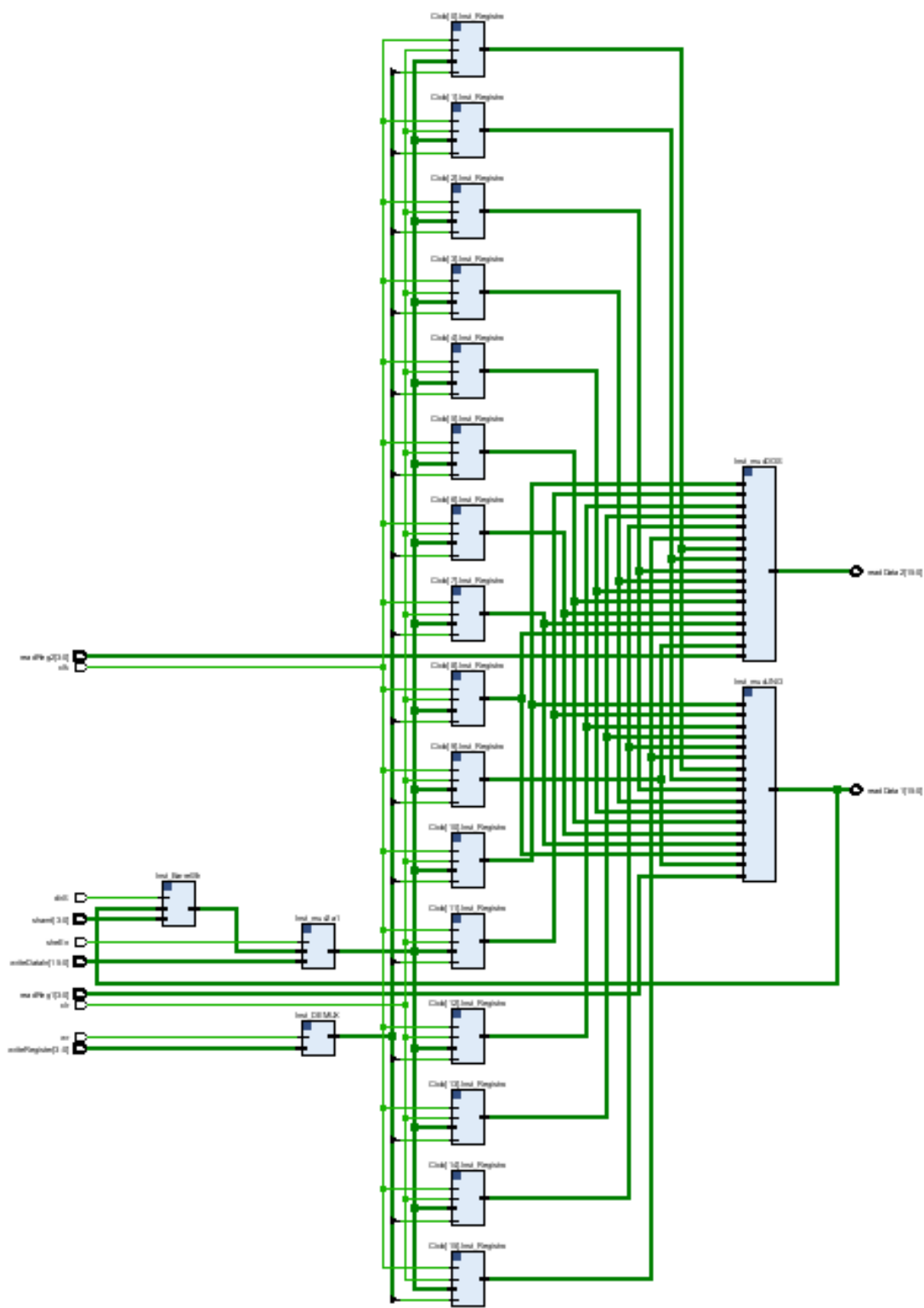
Diagrama RTL Multiplexor 16 Canales**Diagrama RTL BarrelShifter**

Diagrama RTL del Archivo de Registros



4. Simulación Archivo de Registros

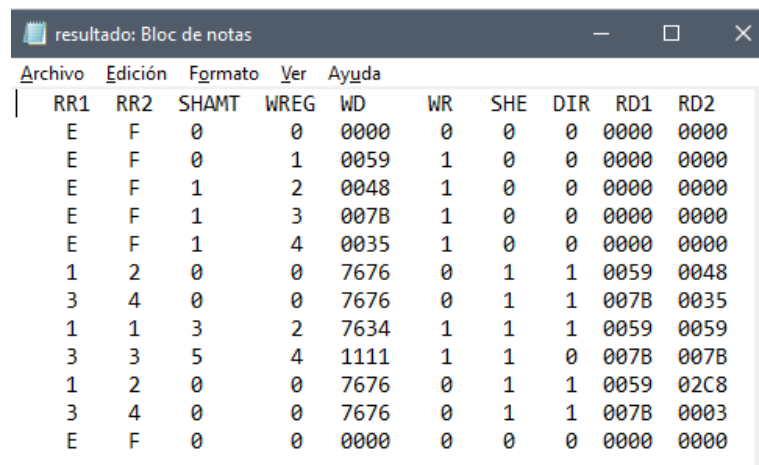
Para realizar la simulación se usó el siguiente conjunto de archivos, siendo el archivo de entrada el nombrado como **entradas.txt** y el de salida el archivo **resultado.txt**; a continuación, se muestra el contenido de ambos.



```

E F 0 0 0000 0 0 0 1
E F 0 1 0059 1 0 0 0
E F 1 2 0048 1 0 0 0
E F 1 3 007B 1 0 0 0
E F 1 4 0035 1 0 0 0
1 2 0 0 7676 0 1 1 0
3 4 0 0 7676 0 1 1 0
1 1 3 2 7634 1 1 1 0
3 3 5 4 1111 1 1 0 0
1 2 0 0 7676 0 1 1 0
3 4 0 0 7676 0 1 1 0
E F 0 0 0000 0 0 0 1
  
```

5. Resultado de la simulación, las salidas del archivo están en formato hexadecimal.



```

RR1  RR2  SHAMT  WREG  WD      WR  SHE  DIR  RD1  RD2
E    F    0      0  0000    0   0   0  0000 0000
E    F    0      1  0059    1   0   0  0000 0000
E    F    1      2  0048    1   0   0  0000 0000
E    F    1      3  007B    1   0   0  0000 0000
E    F    1      4  0035    1   0   0  0000 0000
1    2    0      0  7676    0   1   1  0059 0048
3    4    0      0  7676    0   1   1  007B 0035
1    1    3      2  7634    1   1   1  0059 0059
3    3    5      4  1111    1   1   0  007B 007B
1    2    0      0  7676    0   1   1  0059 02C8
3    4    0      0  7676    0   1   1  007B 0003
E    F    0      0  0000    0   0   0  0000 0000
  
```

6.

a) Código de implementación

Registro

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

entity Registro is
    Port ( d : in STD_LOGIC_VECTOR (15 downto 0);
          q : out STD_LOGIC_VECTOR (15 downto 0);
          clr, clk, l : in STD_LOGIC);
end Registro;

architecture Behavioral of Registro is
begin
    process(clk,clr)
    begin
        if (clr = '1') then
            q <= (others=>'0');
        elsif rising_edge(clk) then
            if(l='1') then
                q <= d;
            end if;
        end if;
    end process;
end Behavioral;
```

Multiplexor 2 canales

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

entity mux2chan is
    Port ( shift : in STD_LOGIC_VECTOR (15 downto 0);
          writeData : in STD_LOGIC_VECTOR (15 downto 0);
          SHE : in STD_LOGIC;
          Dout : out STD_LOGIC_VECTOR (15 downto 0));
end mux2chan;

architecture Behavioral of mux2chan is
begin
    process(SHE, shift, writeData)
    begin
        if SHE = '1' then
            Dout <= shift;
        else
            Dout <= writeData;
        end if;
    end process;
end Behavioral;
```


Multiplexor 16 canal

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

entity mux16chan is
    Port ( readReg : in STD_LOGIC_VECTOR (3 downto 0);
          q0 : in STD_LOGIC_VECTOR (15 downto 0);
          q1 : in STD_LOGIC_VECTOR (15 downto 0);
          q2 : in STD_LOGIC_VECTOR (15 downto 0);
          q3 : in STD_LOGIC_VECTOR (15 downto 0);
          q4 : in STD_LOGIC_VECTOR (15 downto 0);
          q5 : in STD_LOGIC_VECTOR (15 downto 0);
          q6 : in STD_LOGIC_VECTOR (15 downto 0);
          q7 : in STD_LOGIC_VECTOR (15 downto 0);
          q8 : in STD_LOGIC_VECTOR (15 downto 0);
          q9 : in STD_LOGIC_VECTOR (15 downto 0);
          qA : in STD_LOGIC_VECTOR (15 downto 0);
          qB : in STD_LOGIC_VECTOR (15 downto 0);
          qC : in STD_LOGIC_VECTOR (15 downto 0);
          qD : in STD_LOGIC_VECTOR (15 downto 0);
          qE : in STD_LOGIC_VECTOR (15 downto 0);
          qF : in STD_LOGIC_VECTOR (15 downto 0);
          readData : out STD_LOGIC_VECTOR (15 downto 0));
end mux16chan;

architecture Behavioral of mux16chan is
begin
    readData <= q0 when readReg = "0000" else
                q1 when readReg = "0001" else
                q2 when readReg = "0010" else
                q3 when readReg = "0011" else
                q4 when readReg = "0100" else
                q5 when readReg = "0101" else
                q6 when readReg = "0110" else
                q7 when readReg = "0111" else
                q8 when readReg = "1000" else
                q9 when readReg = "1001" else
                qA when readReg = "1010" else
                qB when readReg = "1011" else
                qC when readReg = "1100" else
                qD when readReg = "1101" else
                qE when readReg = "1110" else
                qF when readReg = "1111" else
                q0;

end Behavioral;
```

Demultiplexor

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Demux is
    Port ( WR : in STD_LOGIC;
          writeReg : in STD_LOGIC_VECTOR (3 downto 0);
          Lout : out STD_LOGIC_VECTOR (15 downto 0));
end Demux;

architecture Behavioral of Demux is

begin
    process(WR, writeReg)
    begin
        if WR = '0' then
            Lout <= (others=>'0');
        else
            if writeReg = "0000" then
                Lout <= "0000000000000001";
            elsif writeReg = "0001" then
                Lout <= "0000000000000010";
            elsif writeReg = "0010" then
                Lout <= "0000000000000100";
            elsif writeReg = "0011" then
                Lout <= "0000000000001000";
            elsif writeReg = "0100" then
                Lout <= "0000000000010000";
            elsif writeReg = "0101" then
                Lout <= "0000000000100000";
            elsif writeReg = "0110" then
                Lout <= "0000000001000000";
            elsif writeReg = "0111" then
                Lout <= "0000000010000000";
            elsif writeReg = "1000" then
                Lout <= "0000000100000000";
            elsif writeReg = "1001" then
                Lout <= "0000001000000000";
            elsif writeReg = "1010" then
                Lout <= "0000010000000000";
            elsif writeReg = "1011" then
                Lout <= "0000100000000000";
            elsif writeReg = "1100" then
                Lout <= "0001000000000000";
            elsif writeReg = "1101" then
                Lout <= "0010000000000000";
            elsif writeReg = "1110" then
                Lout <= "0100000000000000";
            else
                Lout <= "1000000000000000";
            end if;
        end if;
    end process;

end Behavioral;

```

BarrelShifter

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

entity barrelSh is
    Port ( readData : in STD_LOGIC_VECTOR (15 downto 0);
          s : in STD_LOGIC_VECTOR (3 downto 0);
          dir : in STD_LOGIC;
          res : out STD_LOGIC_VECTOR (15 downto 0));
end barrelSh;

architecture Behavioral of barrelSh is

begin
    process(dir,s,readData)
        variable aux : std_logic_vector(15 downto 0);
        begin
            aux := readData;
            if dir = '1' then
                for i in 0 to 3 loop
                    for j in 15 downto 2**i loop
                        if s(i) = '0' then
                            aux(j) := aux(j);
                        else
                            aux(j) := aux(j-2**i);
                        end if;
                    end loop;
                    for j in 2**i-1 downto 0 loop
                        if s(i) = '0' then
                            aux(j) := aux(j);
                        else
                            aux(j) := '0';
                        end if;
                    end loop;
                end loop;
            else
                for i in 0 to 3 loop
                    for j in 0 to 15-2**i loop
                        if s(i) = '0' then
                            aux(j) := aux(j);
                        else
                            aux(j) := aux(j+2**i);
                        end if;
                    end loop;
                    for j in 15-2**i+1 to 15 loop
                        if s(i) = '0' then
                            aux(j) := aux(j);
                        else
                            aux(j) := '0';
                        end if;
                    end loop;
                end loop;
            end if;
            res <= aux;
        end process;
    end Behavioral;
```

Archivo de Registros

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

entity ArchivoRegistro is
    generic (
        n : integer := 16
    );
    Port ( readReg1,readReg2 : in  STD_LOGIC_VECTOR (3 downto 0);
          shamt : in  STD_LOGIC_VECTOR (3 downto 0);
          writeRegister : in  STD_LOGIC_VECTOR (3 downto 0);
          writeDataIn : in  STD_LOGIC_VECTOR (n-1 downto 0);
          wr : in  STD_LOGIC;
          sheEn : in  STD_LOGIC;
          dirS : in  STD_LOGIC;
          clk,clr : in  STD_LOGIC;
          readData1,readData2 : inout  STD_LOGIC_VECTOR (n-1 downto 0));
end ArchivoRegistro;

architecture Behavioral of ArchivoRegistro is

    TYPE MATRIX IS ARRAY(0 TO N-1) OF STD_LOGIC_VECTOR(N-1 DOWNT0 0);

    SIGNAL REG_MATRIX : MATRIX;

    SIGNAL L_VECTOR      : STD_LOGIC_VECTOR(N-1 DOWNT0 0);

    SIGNAL NEW_DATA      : STD_LOGIC_VECTOR(N-1 DOWNT0 0);

    SIGNAL BS_WRITE_DATA : STD_LOGIC_VECTOR(N-1 DOWNT0 0);

    COMPONENT Registro
        Port ( d : in STD_LOGIC_VECTOR (15 downto 0);
              q : out STD_LOGIC_VECTOR (15 downto 0);
              clr, clk, l : in STD_LOGIC);
    END COMPONENT;

    COMPONENT Demux
        Port ( WR : in STD_LOGIC;
              writeReg : in STD_LOGIC_VECTOR (3 downto 0);
              Lout : out STD_LOGIC_VECTOR (15 downto 0));
    END COMPONENT;

    COMPONENT barrelSh
        Port ( readData : in STD_LOGIC_VECTOR (15 downto 0);
              s : in STD_LOGIC_VECTOR (3 downto 0);
              dir : in STD_LOGIC;
              res : out STD_LOGIC_VECTOR (15 downto 0));
    END COMPONENT;
```

```
COMPONENT mux16chan
  Port ( readReg : in STD_LOGIC_VECTOR (3 downto 0);
        q0 : in STD_LOGIC_VECTOR (15 downto 0);
        q1 : in STD_LOGIC_VECTOR (15 downto 0);
        q2 : in STD_LOGIC_VECTOR (15 downto 0);
        q3 : in STD_LOGIC_VECTOR (15 downto 0);
        q4 : in STD_LOGIC_VECTOR (15 downto 0);
        q5 : in STD_LOGIC_VECTOR (15 downto 0);
        q6 : in STD_LOGIC_VECTOR (15 downto 0);
        q7 : in STD_LOGIC_VECTOR (15 downto 0);
        q8 : in STD_LOGIC_VECTOR (15 downto 0);
        q9 : in STD_LOGIC_VECTOR (15 downto 0);
        qA : in STD_LOGIC_VECTOR (15 downto 0);
        qB : in STD_LOGIC_VECTOR (15 downto 0);
        qC : in STD_LOGIC_VECTOR (15 downto 0);
        qD : in STD_LOGIC_VECTOR (15 downto 0);
        qE : in STD_LOGIC_VECTOR (15 downto 0);
        qF : in STD_LOGIC_VECTOR (15 downto 0);
        readData : out STD_LOGIC_VECTOR (15 downto 0));
END COMPONENT;

COMPONENT mux2chan
  Port ( shift : in STD_LOGIC_VECTOR (15 downto 0);
        writeData : in STD_LOGIC_VECTOR (15 downto 0);
        SHE : in STD_LOGIC;
        Dout : out STD_LOGIC_VECTOR (15 downto 0));
END COMPONENT;

begin

  Ciclo: FOR i IN 0 TO N-1 GENERATE
    Inst_Registro: Registro PORT MAP(
      clk => clk,
      clr => clr,
      l => L_VECTOR(i),
      d => NEW_DATA,
      q => REG_MATRIX(i)
    );
  END GENERATE Ciclo;
```

```
Inst_muxUN0: mux16chan PORT MAP(  
    readReg => readReg1,  
    q0 => REG_MATRIX(0),  
    q1 => REG_MATRIX(1),  
    q2 => REG_MATRIX(2),  
    q3 => REG_MATRIX(3),  
    q4 => REG_MATRIX(4),  
    q5 => REG_MATRIX(5),  
    q6 => REG_MATRIX(6),  
    q7 => REG_MATRIX(7),  
    q8 => REG_MATRIX(8),  
    q9 => REG_MATRIX(9),  
    qA => REG_MATRIX(10),  
    qB => REG_MATRIX(11),  
    qC => REG_MATRIX(12),  
    qD => REG_MATRIX(13),  
    qE => REG_MATRIX(14),  
    qF => REG_MATRIX(15),  
    readData => readData1  
);  
  
Inst_muxDOS: mux16chan PORT MAP(  
    readReg => readReg2,  
    q0 => REG_MATRIX(0),  
    q1 => REG_MATRIX(1),  
    q2 => REG_MATRIX(2),  
    q3 => REG_MATRIX(3),  
    q4 => REG_MATRIX(4),  
    q5 => REG_MATRIX(5),  
    q6 => REG_MATRIX(6),  
    q7 => REG_MATRIX(7),  
    q8 => REG_MATRIX(8),  
    q9 => REG_MATRIX(9),  
    qA => REG_MATRIX(10),  
    qB => REG_MATRIX(11),  
    qC => REG_MATRIX(12),  
    qD => REG_MATRIX(13),  
    qE => REG_MATRIX(14),  
    qF => REG_MATRIX(15),  
    readData => readData2  
);  
  
Inst_DEMUX: Demux PORT MAP(  
    WR      => wr,  
    writeReg => writeRegister,  
    Lout     => L_VECTOR  
);  
  
Inst_BarrelSh: barrelSh PORT MAP(  
    readData => readData1,  
    s        => shamt,  
    dir      => dirS,  
    res      => BS_WRITE_DATA  
);  
  
Inst_mux2a1: mux2chan PORT MAP(  
    shift     => BS_WRITE_DATA,  
    writeData => writeDataIn,  
    SHE => sheEN,  
    Dout => NEW_DATA  
);  
  
end Behavioral;
```

b) Código de simulación(test bench)**Registro**

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

entity Registro_tb is
end Registro_tb;

architecture Behavioral of Registro_tb is
    component Registro is
        Port ( d : in STD_LOGIC_VECTOR (15 downto 0);
              q : out STD_LOGIC_VECTOR (15 downto 0);
              clr, clk, l : in STD_LOGIC);
    end component;

    signal d : STD_LOGIC_VECTOR (15 downto 0);
    signal q : STD_LOGIC_VECTOR (15 downto 0);
    signal clr, clk, l : STD_LOGIC;

    begin

        u1 : Registro
            Port map (
                d => d,
                q => q,
                clk => clk,
                clr => clr,
                l => l
            );

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

        reg : process
        begin
            clr <= '1';
            wait for 40 ns;
            clr <= '0';
            l <= '1';
            d <= x"1998";
            wait for 10 ns;
            l <= '0';
            d <= x"0042";
            wait;
        end process;

    end Behavioral;
```

Barrel Shifter

```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;

entity barrelSh_tb is
end;

architecture bench of barrelSh_tb is

    component barrelSh
        Port ( readData : in STD_LOGIC_VECTOR (15 downto 0);
              s : in STD_LOGIC_VECTOR (3 downto 0);
              dir : in STD_LOGIC;
              res : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    signal readData: STD_LOGIC_VECTOR (15 downto 0);
    signal s: STD_LOGIC_VECTOR (3 downto 0);
    signal dir: STD_LOGIC;
    signal res: STD_LOGIC_VECTOR (15 downto 0);

begin

    uut: barrelSh port map ( readData => readData,
                             s      => s,
                             dir    => dir,
                             res    => res );

    stimulus: process
    begin

        s <= "1001";
        readData <= x"FFFF";
        dir <= '0';
        wait for 100ns;
        readData <= x"00FF";
        dir <= '1';
        s <= "0101";
        wait for 100ns;
        wait;
    end process;

end;
```


Archivo de Registros

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

USE ieee.std_logic_TEXTIO.ALL;  --PERMITE USAR STD_LOGIC
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_UNSIGNED.ALL;
USE ieee.std_logic_ARITH.ALL;

entity tb_ArchivoRegistro is
end tb_ArchivoRegistro;

architecture tb of tb_ArchivoRegistro is

    component ArchivoRegistro
        port (readReg1      : in std_logic_vector (3 downto 0);
              readReg2      : in std_logic_vector (3 downto 0);
              shamt          : in std_logic_vector (3 downto 0);
              writeRegister  : in std_logic_vector (3 downto 0);
              writeDataIn    : in std_logic_vector (15 downto 0);
              wr              : in std_logic;
              sheEn           : in std_logic;
              dirS            : in std_logic;
              clk             : in std_logic;
              clr             : in std_logic;
              readData1       : inout std_logic_vector (15 downto 0);
              readData2       : inout std_logic_vector (15 downto 0));
    end component;

    signal readReg1      : std_logic_vector (3 downto 0);
    signal readReg2      : std_logic_vector (3 downto 0);
    signal shamt          : std_logic_vector (3 downto 0);
    signal writeRegister  : std_logic_vector (3 downto 0);
    signal writeDataIn    : std_logic_vector (15 downto 0);
    signal wr              : std_logic;
    signal sheEn           : std_logic;
    signal dirS            : std_logic;
    signal clk             : std_logic;
    signal clr             : std_logic;
    signal readData1       : std_logic_vector (15 downto 0);
    signal readData2       : std_logic_vector (15 downto 0);

    constant CLK_period : time := 10 ns;

begin
```

```
dut : ArchivoRegistro
port map (readReg1      => readReg1,
          readReg2      => readReg2,
          shamt          => shamt,
          writeRegister  => writeRegister,
          writeDataIn    => writeDataIn,
          wr             => wr,
          sheEn          => sheEn,
          dirS           => dirS,
          clk            => clk,
          clr            => clr,
          readData1      => readData1,
          readData2      => readData2
        );

CLK_process :process
begin
    CLK <= '0';
    wait for CLK_period/2;
    CLK <= '1';
    wait for CLK_period/2;
end process;

stimuli : process
file ARCH_RES : TEXT;
variable LINEA_RES : LINE;
VARIABLE VAR_RDU : STD_LOGIC_VECTOR(15 DOWNT0 0);
VARIABLE VAR_RDD : STD_LOGIC_VECTOR(15 DOWNT0 0);

file ARCH_VEC : TEXT;
variable LINEA_VEC : line;

VARIABLE VAR_RR1 : STD_LOGIC_VECTOR(3 DOWNT0 0);
VARIABLE VAR_RR2 : STD_LOGIC_VECTOR(3 DOWNT0 0);
VARIABLE VAR_SHAMT : STD_LOGIC_VECTOR(3 DOWNT0 0);
VARIABLE VAR_WRE : STD_LOGIC_VECTOR(3 DOWNT0 0);
VARIABLE VAR_WD : STD_LOGIC_VECTOR(15 DOWNT0 0);
VARIABLE VAR_WR : STD_LOGIC;
VARIABLE VAR_SHE : STD_LOGIC;
VARIABLE VAR_DIR : STD_LOGIC;
VARIABLE VAR_CLR : STD_LOGIC;

VARIABLE CADENA : STRING(1 TO 4);
VARIABLE CADENA_I : STRING(1 TO 6);
VARIABLE CADENA_W : STRING(1 TO 10);
VARIABLE CADENA_X : STRING(1 TO 5);
```

```
begin
    file_open(ARCH_RES, "D:\ESCOM\ARQUITECTURA\ArchivoRegistros\Archi
voRegistros.srcs\sim_1\new\resultado.txt", WRITE_MODE);
    file_open(ARCH_VEC, "D:\ESCOM\ARQUITECTURA\ArchivoRegistros\Archi
voRegistros.srcs\sim_1\new\entradas.txt", READ_MODE);

    CADENA_X := " RR1";
    write(LINEA_RES, CADENA_X, right, CADENA_X'LENGTH+1);
    CADENA := " RR2";
    write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
    CADENA_I := " SHAMT";
    write(LINEA_RES, CADENA_I, right, CADENA_I'LENGTH+1);
    CADENA_X := " WREG";
    write(LINEA_RES, CADENA_X, right, CADENA_X'LENGTH+1);
    CADENA := " WD ";
    write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
    CADENA_I := " WR ";
    write(LINEA_RES, CADENA_I, right, CADENA_I'LENGTH+1);
    CADENA := " SHE";
    write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
    CADENA := " DIR";
    write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
    CADENA := " RD1";
    write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
    CADENA := " RD2";
    write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
    writeline(ARCH_RES, LINEA_RES);
    wait for 100 ns;

    FOR I IN 0 TO 11 LOOP
        readline(ARCH_VEC, LINEA_VEC);

        Hread(LINEA_VEC, VAR_RR1);
        readReg1<=VAR_RR1;

        Hread(LINEA_VEC, VAR_RR2);
        readReg2<=VAR_RR2;

        Hread(LINEA_VEC, VAR_SHAMT);
        SHAMT<=VAR_SHAMT;

        Hread(LINEA_VEC, VAR_WRE);
        writeRegister<=VAR_WRE;

        Hread(LINEA_VEC, VAR_WD);
        writeDataIn<=VAR_WD;
```

```
    read(LINEA_VEC, VAR_WR);
    WR<=VAR_WR;

    read(LINEA_VEC, VAR_SHE);
    sheEn<=VAR_SHE;

    read(LINEA_VEC, VAR_DIR);
    dirS<=VAR_DIR;

    read(LINEA_VEC, VAR_CLR);
    CLR<=VAR_CLR;

    WAIT UNTIL RISING_EDGE(CLK);
    VAR_RDU := readData1;
    VAR_RDD := readData2;

    Hwrite(LINEA_RES, VAR_RR1, right, 5);
    Hwrite(LINEA_RES, VAR_RR2, right, 5);
    Hwrite (LINEA_RES, VAR_SHAMT, right, 5);
    Hwrite (LINEA_RES, VAR_WRE,      right, 8);
    Hwrite (LINEA_RES, VAR_WD, right, 7);
    write(LINEA_RES, VAR_WR,      right, 5);
    write(LINEA_RES, VAR_SHE,      right, 5);
    write(LINEA_RES, VAR_DIR,      right, 5);
    Hwrite (LINEA_RES, VAR_RDU, right, 6);
    Hwrite (LINEA_RES, VAR_RDD, right, 6);
    writeline(ARCH_RES, LINEA_RES);

end loop;

file_close(ARCH_VEC);
file_close(ARCH_RES);

wait for CLK_period*10;

wait;
end process;

end tb;
```