Instituto Politécnico Nacional

Escuela Superior de Computo

Práctica 10.

Ruta de datos del ESCOMips.

Nombre: Flores Castro Luis Antonio.

Arquitectura de Computadoras.

Profesora: Vega García Nayeli.

Código de paquete de la Ruta de Datos.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. package paqueteRD is
5.
6. component PilaH is
7.     Port ( pcIn : in STD_LOGIC_VECTOR (15 downto 0);
8.            clk, clr, wPC, UP, DW : in STD_LOGIC;
9.            pcOut : out STD_LOGIC_VECTOR (15 downto 0));
10.     end component;
11.
12.     --memoria de programa
13.     component dataprogram is
14.         Port ( PC : in STD_LOGIC_VECTOR (9 downto 0);
15.                ints : out STD_LOGIC_VECTOR (24 downto 0));
16.     end component;
17.
18.     --Archivo de Registros
19.     component FileReg is
20.             Port( writeReg, readReg1, readReg2,
   shamt: in STD_LOGIC_VECTOR(3 downto 0);
21.                   writeData: in STD_LOGIC_VECTOR(15 downto
    0);
22.                   clk,clr,wr,she,dir: in STD_LOGIC;
23.                   readData1,
   readData2: out STD_LOGIC_VECTOR(15 downto 0));
24.     end component;
25.
26.     --ALU
27.     component ALU4bits is
28.     Port( a : in STD_LOGIC_VECTOR(15 downto 0);
29.             b : in STD_LOGIC_VECTOR(15 downto 0);
30.             Aluop : in STD_LOGIC_VECTOR(3 downto 0);
31.             Res : inout STD_LOGIC_VECTOR(15 downto 0);
32.             bandera : out STD_LOGIC_VECTOR(3 downto 0);
33.             Cout : out STD_LOGIC
34.             );
35.     end component;
36.
37.     --Memoria de Datos
38.
39.     component datamemory is
40.             Port(dir:in STD_LOGIC_VECTOR(9 downto 0);
41.             dataIn: in STD_LOGIC_VECTOR(15 downto 0);
42.             WD, clk: STD_LOGIC;
43.             dataOut: out STD_LOGIC_VECTOR(15 downto 0));
44.     end component;
45.
46.     --Extensor de Signo
```

```
47.     component ExtensorSigno is
48.             Port(sliti: in STD_LOGIC_VECTOR(11 downto 0);
49.                     slito: out STD_LOGIC_VECTOR(15 downto 0)
   );
50.     end component;
51.
52.     --Extensor de Direccion
53.     component ExtensorDireccion is
54.     Port( dliti: in STD_LOGIC_VECTOR(11 downto 0);
55.             dlito: out STD_LOGIC_VECTOR(15 downto 0));
56.     end component;
57.
58.     --mux de 4 bits
59.     component mux4b is
60.     Port(A,B: in std_logic_vector(3 downto 0);
61.             selector: in STD_LOGIC;--SOLO UN BIT
62.             salida: out STD_LOGIC_VECTOR(3 downto 0));
63.     end component;
64.
65.     --mux de 16 bits
66.     component mux16b is
67.     Port(A,B: in std_logic_vector(15 downto 0);
68.             selector: in STD_LOGIC;--SOLO UN BIT
69.             salida: out STD_LOGIC_VECTOR(15 downto 0));
70.     end component;
71.
72.
73.
74.     component muxSR2 is
75.         Port ( A, B : in STD_LOGIC_VECTOR (3 downto 0);
76.                selector : in STD_LOGIC;
77.                salida : out STD_LOGIC_VECTOR (3 downto 0));
78.     end component;
79.
80.     end package;
```

Código de Implementación de la Ruta de Datos.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use work.paqueteRD.all;
4.
5. entity RutaDatos is
6.         Port(rdclr, clk: in STD_LOGIC);
7.
8. end RutaDatos;
9.
10.     architecture Behavioral of RutaDatos is
11.
12.     signal AUXSDMP, AUXSWD, AUXSEXT, AUXSOP1, AUXSOP2,
   AUXSDMD, AUXSR,
   AUXPCOUT:STD_LOGIC_VECTOR(15 downto 0):=(others=>'0');
13.     signal AUXINST:STD_LOGIC_VECTOR(24 downto 0):=(others=>'
   0');
14.     signal clr: STD_LOGIC:='0';
15.     signal AUXBANDERAS,
   AUXSR2: STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
16.     signal AUXINSTRUCCION: STD_LOGIC_VECTOR(19 downto 0):=(o
   thers=>'0');
17.     signal AUXREADDATA1,
   AUXREADDATA2: STD_LOGIC_VECTOR(15 downto 0):=(others=>'0');
18.     signal AUXEXTDIR, AUXEXTSIG, AUXRESALU,
   AUXDATAOUT: STD_LOGIC_VECTOR(15 downto 0):=(others=>'0');
19.
20.     begin
21.
22.     process(clk)
23.     begin
24.     if(falling_edge(clk))then
25.             clr<=rdclr;
26.     end if;
27.     end process;
28.     --AQUI VAN LAS CONEXIONES YA
29.
30.     --entidad para pila
31.     sP: PilaH
32.     Port map(
33.     pcIn=>AUXSDMP,
34.     clk=>clk,
35.     clr=>clr,
36.     wPC=>AUXINSTRUCCION(16),
37.     UP=>AUXINSTRUCCION(18),
38.     DW=>AUXINSTRUCCION(17),
39.     pcOut=>AUXPCOUT
40.     );
41.
42.     --CONEXION MEMORIA DE PROGRAMA
```

```
43.     mpro: dataprogram
44.     Port map(
45.     PC=>AUXPCOUT(9 downto 0),
46.     ints=>AUXINST
47.     );
48.
49.     --multiplexor SR2
50.     SR2: muxsr2
51.     Port map(
52.     A=>AUXINST(11 downto 8),
53.     B=>AUXINST(19 downto 16),
54.     selector=>AUXINSTRUCCION(15),
55.     salida=>AUXSR2
56.     );
57.
58.     --MULTIPLEXOR SWD
59.     SWD: mux16b
60.     Port map(
61.     A=>AUXINST(15 downto 0),
62.     B=>AUXSR,
63.     selector=>AUXINSTRUCCION(7),
64.     salida=>AUXSWD
65.     );
66.
67.     --CONEXION DE ARCHIVO DE REGISTROS
68.     AR: FileReg
69.     Port map(
70.     wr=>AUXINSTRUCCION(10),
71.     dir=>AUXINSTRUCCION(11),
72.     she=>AUXINSTRUCCION(12),
73.     clk=>clk,
74.     clr=>clr,
75.     writeReg=>AUXINST(19 downto 16),
76.     readReg1=>AUXINST(15 downto 12),
77.     readReg2=>AUXSR2,
78.     shamt=>AUXINST(7 downto 4),
79.     writeData=>AUXSWD,
80.     readData1=>AUXREADDATA1,
81.     readData2=>AUXREADDATA2
82.     );
83.
84.     --CONEXION DE EXTENSORES
85.     ESIGNO: ExtensorSigno
86.     Port map(
87.     sliti=>AUXINST(11 downto 0),
88.     slito=>AUXEXTSIG
89.     );
90.
91.     EDIR : ExtensorDireccion
92.     Port map(
93.     dliti=>AUXINST(11 downto 0),
94.     dlito=>AUXEXTDIR
```

```
95.     );
96.
97.     --MULTIPLEXOR SEXT UNIDAD A EXTENSORES
98.     SEXT: mux16b
99.     Port map(
100.    A=>AUXEXTSIG,
101.    B=>AUXEXTDIR,
102.    selector=>AUXINSTRUCCION(13),
103.    salida=>AUXSEXT
104.    );
105.
106.    --MULTIPLEXOR SOP1 Y SOP2, UNIDAD A ARCHIVO DE REGISTROS
   Y ALU
107.    SOP1: mux16b
108.    Port map(
109.    A=>AUXREADDATA1,
110.    B=>AUXPCOUT,
111.    selector=>AUXINSTRUCCION(9),
112.    salida=>AUXSOP1
113.    );
114.
115.    --MULTIPLEXOR SOP1 UNIDAD A ARCHIVO DE REGISTROS Y ALU
116.    SOP2: mux16b
117.    Port map(
118.    A=>AUXREADDATA2,
119.    B=>AUXSEXT,
120.    selector=>AUXINSTRUCCION(8),
121.    salida=>AUXSOP2
122.    );
123.
124.    --CONEXION DE ALU
125.
126.    ALU: ALU4bits
127.    Port map( a=>AUXSOP1,
128.            b=>AUXSOP2,
129.            Aluop=>AUXINSTRUCCION(7 downto 4), --
   AUXINSTRUCCION(7 downto 4),--conexion de unidad de control
   pendiente
130.            Res=>AUXRESALU,
131.            bandera=>AUXBANDERAS);
132.            --pendiente la salida COUT);
133.
134.    --MULTIPLEXOR SDMD A MEMORIA DE DATOS
135.    SDMD: mux16b
136.    Port map(
137.    A=>AUXRESALU,
138.    B=>AUXINST(15 downto 0),
139.    selector=>AUXINSTRUCCION(3),
140.    salida=>AUXSDMD
141.    );
142.
143.    --CONEXION MEMORIA DE DATOS
```

```
144.    mdata: datamemory
145.    Port map(
146.    dir=>AUXSDMD(9 downto 0),
147.    dataIn=>AUXREADDATA2,
148.    WD=>AUXINSTRUCCION(2),
149.    clk=>clk,
150.    dataOut=>AUXDATAOUT
151.    );
152.
153.    --MULTIPLEXOR SR SALIDA DE MEMORIA
154.    SR: mux16b
155.    Port map(
156.    A=>AUXDATAOUT,
157.    B=>AUXRESALU,
158.    selector=>AUXINSTRUCCION(1),
159.    salida=>AUXSR
160.    );
161.
162.    --MULTIPLEXOR SMDP
163.    SDMP: mux16b
164.    Port map(
165.    A=>AUXINST(15 downto 0),
166.    B=>AUXSR,
167.    selector=>AUXINSTRUCCION(19),
168.    salida=>AUXSDMP
169.    );
170.
171.    end Behavioral;
```

Diagrama RTL de la Ruta de Datos.

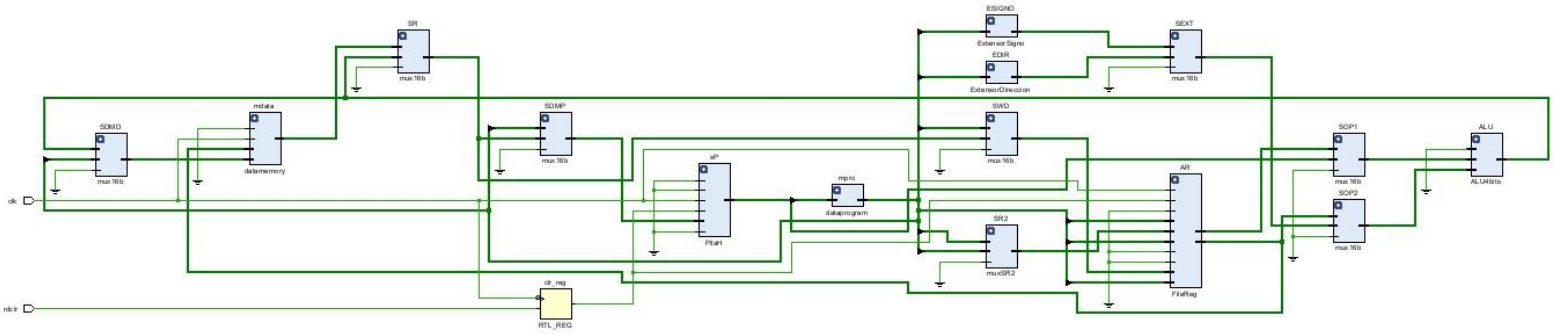

Figura 1. Diagrama RTL de la Ruta de Datos.