Reporte ESCOMips Parte 1

González Pardo Adrian

Junio 2020

1. Código de modulos de hardware

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

package modulosHardware is
    -- Elementos de la ALU 16 Bits
    component sumador is
    Port ( a,b,cin : in STD_LOGIC;
            s,cout : out STD_LOGIC);
    end component;

    component alu is
    Port ( a, b, sela, selb, cin : in STD_LOGIC;
            res,cout : out STD_LOGIC;
            op : in STD_LOGIC_VECTOR (1 downto 0));
    end component;

    --Fin elementos de la ALU

    --Archivo de registros
    component demuxG is
    Port ( input : in STD_LOGIC;
            selector : in STD_LOGIC_VECTOR (3 downto 0);
            output : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

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

    component muxGral is
    Port ( chanel_0,chanel_1,chanel_2,chanel_3,chanel_4,chanel_5,chanel_6,chanel_7,chanel_8
    ,chanel_9,chanel_10,chanel_11,chanel_12,chanel_13,chanel_14,chanel_15 : in
    STD_LOGIC_VECTOR (15 downto 0);
            selectMux : in STD_LOGIC_VECTOR (3 downto 0);
            outMux : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    component barrel is
    Port ( dato : in STD_LOGIC_VECTOR (15 downto 0);
            s : in STD_LOGIC_VECTOR (3 downto 0);
            dir : std_logic;
            res : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    component mux2 is
    Port ( p,s : in STD_LOGIC_VECTOR (15 downto 0);
            sel : in STD_LOGIC;
            sout : out STD_LOGIC_VECTOR (15 downto 0));
    end component;
```

```vhdl
50
51      --Fin de archivo de registros
52
53      --Unidad de control
54      component regControl is
55      Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
56              Q : out STD_LOGIC_VECTOR (3 downto 0);
57              CLR,CLK,L : in STD_LOGIC);
58      end component;
59
60      component condition is
61      Port ( Q : in STD_LOGIC_VECTOR (3 downto 0);
62              EQ,NE,LT,LE,GT,GE : out STD_LOGIC);
63      end component;
64
65      component decInstrucciones is
66      Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
67              TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI : out STD_LOGIC);
68      end component;
69
70      component level is
71      Port ( CLR,CLK : in STD_LOGIC;
72              NA : out STD_LOGIC);
73      end component;
74
75      component mopcode is
76      Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
77              micro : out STD_LOGIC_VECTOR (19 downto 0));
78      end component;
79
80      component mfuncode is
81      Port ( funCode : in STD_LOGIC_VECTOR (3 downto 0);
82              micro : out STD_LOGIC_VECTOR (19 downto 0));
83      end component;
84
85      component unControl is
86      Port ( TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI,NA,EQ,NE,LT,LE,GT,GE : in STD_LOGIC;
87              SM,SDOPC : out STD_LOGIC);
88      end component;
89
90      --Fin Unidad de control
91
92
93
94
95
96
97      --Componentes ESCOMips
98      component aluN is
99      generic(m:integer:=16);
100     Port ( a, b : in STD_LOGIC_VECTOR (m-1 downto 0);
101             aluop : in STD_LOGIC_VECTOR (3 downto 0);
102             res : out STD_LOGIC_VECTOR (m-1 downto 0);
103             n,z,ov,co : out STD_LOGIC
104             );
105     end component;
106
107     --Memoria de datos para almacenamiento
108     component memoriaDatos is
109     generic (
110         m : integer := 16; --tamanio del bus de direcciones
111         n : integer := 16 --tamanio de palabra (dato)
112     );
113     Port(add : in STD_LOGIC_VECTOR (m-1 downto 0);
114         dataIn : in STD_LOGIC_VECTOR (n-1 downto 0);
115         wd, clk : in STD_LOGIC;
116         dataOut : out STD_LOGIC_VECTOR (n-1 downto 0));
```

2

```vhdl
117        end component;
118        --Fin memoria de datos
119        component archRegistros is
120        Port ( writeData : in STD_LOGIC_VECTOR (15 downto 0);
121               writeReg,readReg1,readReg2,shamt : in STD_LOGIC_VECTOR (3 downto 0);
122               WR,CLK,SHE,DIR,clr : in STD_LOGIC;
123               readData1 : inout STD_LOGIC_VECTOR (15 downto 0);
124               readData2 : out STD_LOGIC_VECTOR (15 downto 0));
125        end component;
126        --Memoria de Programa (Contador de programa)
127        component memoriaPrograma is
128        generic(
129            m:integer:= 10;--tamanio del bus de direcciones
130            n:integer:= 25--tamanio de palabra
131        );
132        Port(dir:in STD_LOGIC_VECTOR (m-1 downto 0);
133             inst:out STD_LOGIC_VECTOR (n-1 downto 0));
134        end component;
135        --Fin Memoria de Programa
136        --Pila
137        component pila is
138        generic(
139            N:integer:=16;
140            B:integer:=3
141        );
142        Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
143             DW, UP, WPC, CLK,CLR:in STD_LOGIC;
144             PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
145             SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
146        end component;
147        --Fin Pila
148        component unidadControl is
149        Port ( funCode,banderas : in STD_LOGIC_VECTOR (3 downto 0);
150               CLK,CLR : in STD_LOGIC;
151               opCode : in STD_LOGIC_VECTOR (4 downto 0);
152               microinstruccion : out STD_LOGIC_VECTOR (19 downto 0));
153        end component;
154
155        component extendDir is
156        Port ( entrada : in STD_LOGIC_VECTOR (11 downto 0);
157               salida : out STD_LOGIC_VECTOR (15 downto 0));
158        end component;
159
160        component extendSig is
161        Port ( entrada : in STD_LOGIC_VECTOR (11 downto 0);
162               salida : out STD_LOGIC_VECTOR (15 downto 0));
163        end component;
164
165        component mux4bits is
166        Port ( entrada,entrada1 : in STD_LOGIC_VECTOR (3 downto 0);
167               flag : in STD_LOGIC;
168               salida : out STD_LOGIC_VECTOR (3 downto 0));
169        end component;
170
171        component mux16bits is
172        Port ( entrada,entrada1 : in STD_LOGIC_VECTOR(15 downto 0);
173            flag : in STD_LOGIC;
174            salida : out STD_LOGIC_VECTOR(15 downto 0));
175        end component;
176
177        --Fin ESCOMips
178
179        --ESCOMips
180
181        component escomips is
182        Port ( clk,rclr : in STD_LOGIC;
183            PC : out STD_LOGIC_VECTOR(15 downto 0);
```

```
184          instruccion : out STD_LOGIC_VECTOR(24 downto 0);
185          readData1,readData2,aluOut,busSR : out STD_LOGIC_VECTOR(15 downto 0)
186          );
187      end component;
188      --Fin
189
190  end modulosHardware;
```

2. Código de la arquitectura ESCOMips

```
 1  library IEEE;
 2  library WORK;
 3  use IEEE.STD_LOGIC_1164.ALL;
 4  use WORK.modulosHardware.pila;
 5  use WORK.modulosHardware.memoriaPrograma;
 6  use WORK.modulosHardware.mux4bits;
 7  use WORK.modulosHardware.mux16bits;
 8  use WORK.modulosHardware.extendDir;
 9  use WORK.modulosHardware.extendSig;
10  use WORK.modulosHardware.archRegistros;
11  use WORK.modulosHardware.aluN;
12  use WORK.modulosHardware.memoriaDatos;
13  use WORK.modulosHardware.unidadControl;
14
15  entity escomips is
16      Port ( clk,rclr : in STD_LOGIC;
17          PC : out STD_LOGIC_VECTOR(15 downto 0);
18          instruccion : out STD_LOGIC_VECTOR(24 downto 0);
19          readData1,readData2,aluOut,busSR : out STD_LOGIC_VECTOR(15 downto 0)
20          );
21  end escomips;
22
23  architecture Behavioral of escomips is
24      signal sr2Out,flagsALU : STD_LOGIC_VECTOR(3 downto 0);
25      signal sdmpOut,swdOut,sop1Out,sop2Out,sdmdOut,srOut,sextOut,eSig,eDir :
       STD_LOGIC_VECTOR(15 downto 0);
26      signal outPila : STD_LOGIC_VECTOR(15 downto 0);
27      signal SDMP,DW,UP,WPC,SR2,SWD,SEXT,SHE,DIR,WR,SOP1,SOP2,SDMD,WD,SR,CLR : STD_LOGIC;
28      signal sp : STD_LOGIC_VECTOR(2 downto 0);
29      signal outMemoria : STD_LOGIC_VECTOR(24 downto 0);
30      signal microinstruccion : STD_LOGIC_VECTOR (19 downto 0);
31      signal outReadData1,outReadData2,outALU,outMemDatos : STD_LOGIC_VECTOR(15 downto 0);
32  begin
33
34      process(SDMP,DW,UP,WPC,SR2,SWD,SEXT,SHE,DIR,WR,SOP1,SOP2,SDMD,WD,SR,microinstruccion)
35      begin
36          SDMP<=microinstruccion(19);
37          UP<=microinstruccion(18);
38          DW<=microinstruccion(17);
39          WPC<=microinstruccion(16);
40          SR2<=microinstruccion(15);
41          SWD<=microinstruccion(14);
42          SEXT<=microinstruccion(13);
43          SHE<=microinstruccion(12);
44          DIR<=microinstruccion(11);
45          WR<=microinstruccion(10);
46          SOP1<=microinstruccion(9);
47          SOP2<=microinstruccion(8);
48          SDMD<=microinstruccion(3);
49          WD<=microinstruccion(2);
50          SR<=microinstruccion(1);
51      end process;
52
53      pilaCom:pila port map(
54          PCin=>sdmpOut,
```

4

```vhdl
55          DW=>DW,
56          UP=>UP,
57          WPC=>WPC,
58          CLK=>CLK,
59          CLR=>CLR,
60          PCout=>outPila,
61          SP_out=>sp
62      );
63
64      memoriaProgramaCom:memoriaPrograma port map(
65          dir=>outPila(9 downto 0),
66          inst=>outMemoria
67      );
68
69      sr2Com:mux4bits port map(
70          entrada=>outMemoria(11 downto 8),
71          entrada1=>outMemoria(19 downto 16),
72          flag=>SR2,
73          salida=>sr2Out
74      );
75
76      swdCom:mux16bits port map(
77          entrada=>outMemoria(15 downto 0),
78          entrada1=>srOut,
79          flag=>SWD,
80          salida=>swdOut
81      );
82
83      sdmpCom:mux16bits port map(
84          entrada=>outMemoria(15 downto 0),
85          entrada1=>srOut,
86          flag=>SDMP,
87          salida=>sdmpOut
88      );
89
90      extendDirCom:extendDir port map(
91          entrada=>outMemoria(11 downto 0),
92          salida=>eDir
93      );
94
95      extendSigCom:extendSig port map(
96          entrada=>outMemoria(11 downto 0),
97          salida=>eSig
98      );
99
100     sextCom:mux16bits port map(
101         entrada=>eSig,
102         entrada1=>eDir,
103         flag=>SEXT,
104         salida=>sextOut
105     );
106
107     archRegistrosCom:archRegistros port map(
108         writeData=>swdOut,
109         writeReg=>outMemoria(19 downto 16),
110         readReg1=>outMemoria(15 downto 12),
111         readReg2=>sr2Out,
112         shamt=>outMemoria(7 downto 4),
113         WR=>WR,
114         CLK=>CLK,
115         SHE=>SHE,
116         DIR=>DIR,
117         clr=>CLR,
118         readData1=>outReadData1,
119         readData2=>outReadData2
120     );
121
```

```vhdl
122     sop1Com:mux16bits port map(
123         entrada=>outReadData1,
124         entrada1=>outPila,
125         flag=>SOP1,
126         salida=>sop1Out
127     );
128
129     sop2Com:mux16bits port map(
130         entrada=>outReadData2,
131         entrada1=>sextOut,
132         flag=>SOP2,
133         salida=>sop2Out
134     );
135
136     aluNCom:aluN port map(
137         a=>sop1Out,
138         b=>sop2Out,
139         aluop=>microinstruccion(7 downto 4),
140         res=>outALU,
141         n=>flagsALU(0),
142         z=>flagsALU(2),
143         ov=>flagsALU(3),
144         co=>flagsALU(1)
145     );
146
147     sdmdCom:mux16bits port map(
148         entrada=>outALU,
149         entrada1=>outMemoria(15 downto 0),
150         flag=>SDMD,
151         salida=>sdmdOut
152     );
153
154     memoriaDatosCom:memoriaDatos port map(
155         add=>sdmdOut,
156         dataIn=>outReadData2,
157         WD=>WD,
158         CLK=>CLK,
159         dataOut=>outMemDatos
160     );
161
162     srCom:mux16bits port map(
163         entrada=>outMemDatos,
164         entrada1=>outALU,
165         flag=>SR,
166         salida=>srOut
167     );
168
169     unidadControlCom:unidadControl port map(
170         funCode=>outMemoria(3 downto 0),
171         banderas=>flagsALU,
172         CLK=>CLK,
173         CLR=>CLR,
174         opCode=>outMemoria(24 downto 20),
175         microinstruccion=>microinstruccion
176     );
177
178     process(outMemoria,outReadData1,outReadData2,outAlu,srOut)
179     begin
180         PC<=outPila;
181         instruccion<=outMemoria;
182         readData1<=outReadData1;
183         readData2<=outReadData2;
184         aluOut<=outAlu;
185         busSR<=srOut;
186     end process;
187
188     process(CLK)
```

6

```vhdl
      begin
          if falling_edge(CLK) then
              CLR<=RCLR;
          end if;
      end process;


end Behavioral;
```

3. RTL de la arquitectura
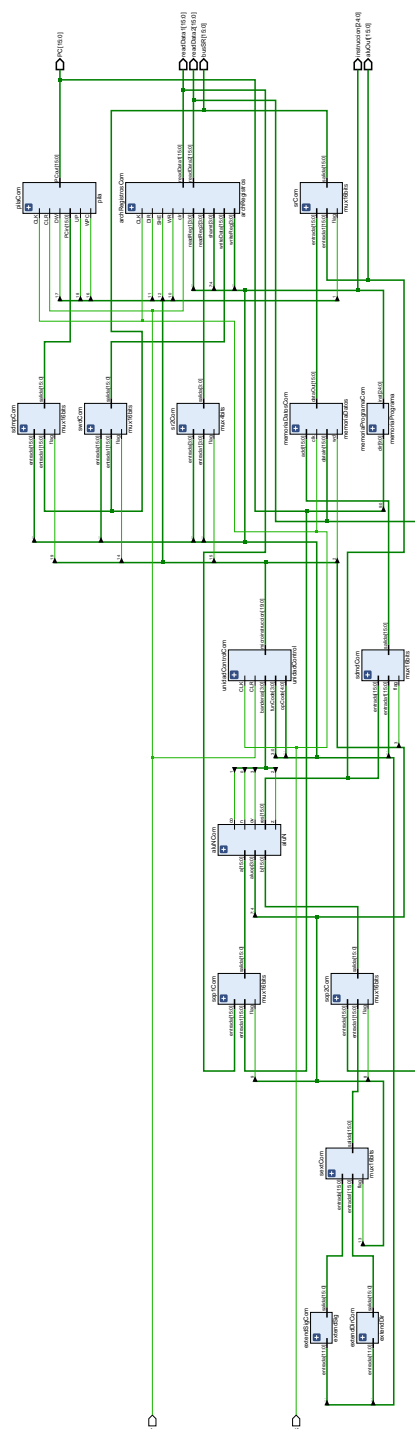
4. Código del primer programa en Memoria de programa

```vhdl
type banco is array (0 to (2**m)-1) of std_logic_vector(n-1 downto 0);
constant aux:banco:=(
  -- Primer programa proyecto
  LI & R0 & x"0001" ,               -- 0 LI R0, #1
  LI & R1 & x"0007" ,               -- 1 LI R1, #7
  TYPER & R1 & R1 & R0 & SU & ADD , -- 2 ADD R1,R1,R0
  SWI & R1 & x"0005" ,              -- 3 SWI R1,#5
  B& SU & x"0002" ,                 -- 4 B 2
  others=>(others=>'0')
  );
```

5. Simulación imagenes

6. Tabla del programa 1

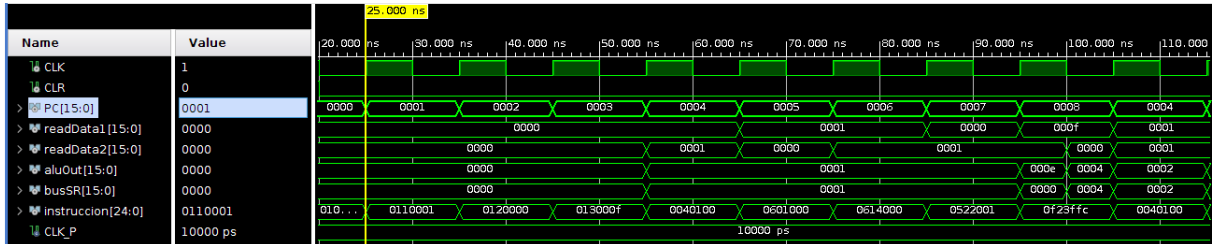| Bus	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
PC	0000	0001	0002	0003	0004	0002	0003	0004	0002	0003	0004
Instrucción	0100001	0110007	0011000	0310005	1300002	0011000	0310005	1300002	0011000	0310005	1300002
Read Data 1	0000	0001	0007	0001	0001	0008	0001	0001	0009	0001	0001
Read Data 2	0000	0001	0001	0008	0001	0001	0009	0001	0001	000a	0001
Res ALU	0000	0001	0008	0000	0001	0009	0001	0001	000a	0000	0001
Bus SR	0000	0000	0008	0000	0000	0009	0008	0000	000a	0009	0000

7. Código del segundo programa en Memoria de programa

```vhdl
type banco is array (0 to (2**m)-1) of std_logic_vector(n-1 downto 0);

constant aux:banco:=(
    -- Segundo programa proyecto Fibonacci
    LI & R0 & x"0000" ,                 -- 0 LI R0, #0
    LI & R1 & x"0001" ,                 -- 1 LI R1, #1
    LI & R2 & x"0000" ,                 -- 2 LI R2, #0
    LI & R3 & x"000F" ,                 -- 3 LI R3, #15
    TYPER & R4 & R0 & R1 & SU & ADD ,   -- 4 ADD R1,R1,R0
    SUBI & R0 & R1 & x"000" ,           -- 5 SUBI R0,R1,#0
    SUBI & R1 & R4 & x"000" ,           -- 6 SUBI R1,R4,#0
    ADDI & R2 & R2 & x"001" ,           -- 7 ADDI R2,R2,#1
    BLTI & R2 & R3 & x"FFC" ,           -- 8 BLTI R3,R2,-4
    NOP & SU & SU & SU & SU & SU ,      -- 9 NOP
    B & SU & x"0009" ,                  --10 B 9
    others=>(others=>'0')
);
```

8. Simulación imagenes

9. Tabla del programa 2

Bus	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
PC	0000	0001	0002	0003	0004	0005	0006	0007	0008	0004	0005
Instrucción	0100000	0110001	0120000	013000f	0040100	0601000	0614000	0522001	0f23ffc	0040100	0601000
Read Data 1	0000	0000	0000	0000	0000	0001	0001	0000	0001	0001	0001
Read Data 2	0000	0000	0000	0000	0001	0000	0001	0001	000F&0000	0001	0001
Res ALU	0000	0000	0000	0000	0001	0001	0001	0001	FFF2&0004	0002	0001
Bus SR	0000	0000	0000	0000	0001	0001	0001	0001	0000&0004	0002	0001