Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Implementación del RV32IM monociclo en VHDL

Autor:

Mendoza González, Jorge

Tutor(es):

Cáceres Gómez, Santiago

Departamento de Tecnología Electrónica

Valladolid, Julio 2022.

Índice de los Anexos

Índice de las figuras

Anexo 1: Estándares de la codificación de las instrucciones que realiza el procesador RISC-V

Para la generación de las instrucciones empleadas en las pruebas de la arquitectura RISC-V se ha usado los estándares de instrucción que se muestran en la Ilustración 1 y la Ilustración 2.
En la Ilustración 1 se observan todos estándares para las instrucciones básicas de la arquitectura RISC-V. En la Ilustración 2 se ven los estándares para las instrucciones de la extensión para las operaciones de multiplicación y división.

31 25	24 20	19 15	14 12	11 7	6 0	
imm[31:12]				rd	0110111	U lui
imm[31:12]				rd	0010111	U auipc
imm[20|10:1|11|19:12]				rd	1101111	J jal
imm[11:0]		rs1	000	rd	1100111	I jalr
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	B beq
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	B bne
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	B blt
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	B bge
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	B bltu
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	B bgeu
imm[11:0]		rs1	000	rd	0000011	I lb
imm[11:0]		rs1	001	rd	0000011	I lh
imm[11:0]		rs1	010	rd	0000011	I lw
imm[11:0]		rs1	100	rd	0000011	I lbu
imm[11:0]		rs1	101	rd	0000011	I lhu
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	S sb
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	S sh
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	S sw
imm[11:0]		rs1	000	rd	0010011	I addi
imm[11:0]		rs1	010	rd	0010011	I slti
imm[11:0]		rs1	011	rd	0010011	I sltiu
imm[11:0]		rs1	100	rd	0010011	I xori
imm[11:0]		rs1	110	rd	0010011	I ori
imm[11:0]		rs1	111	rd	0010011	I andi
0000000	shamt	rs1	001	rd	0010011	I slli
0000000	shamt	rs1	101	rd	0010011	I srli
0100000	shamt	rs1	101	rd	0010011	I srai
0000000	rs2	rs1	000	rd	0110011	R add
0100000	rs2	rs1	000	rd	0110011	R sub
0000000	rs2	rs1	001	rd	0110011	R sll
0000000	rs2	rs1	010	rd	0110011	R slt
0000000	rs2	rs1	011	rd	0110011	R sltu
0000000	rs2	rs1	100	rd	0110011	R xor
0000000	rs2	rs1	101	rd	0110011	R srl
0100000	rs2	rs1	101	rd	0110011	R sra
0000000	rs2	rs1	110	rd	0110011	R or
0000000	rs2	rs1	111	rd	0110011	R and

Ilustración 1: El mapa de opcodes de RV32I tiene la estructura de la instrucción, opcodes, tipo de formato y nombres.[1, p. 2.3]

Anexo 1: Estándares de la codificación de las instrucciones que realiza el procesador RISC-V

31 25	24 20	19 15	14 12	11 7	6 0	
0000001	rs2	rs1	000	rd	0110011	R mul
0000001	rs2	rs1	001	rd	0110011	R mulh
0000001	rs2	rs1	010	rd	0110011	R mulhsu
0000001	rs2	rs1	011	rd	0110011	R mulhu
0000001	rs2	rs1	100	rd	0110011	R div
0000001	rs2	rs1	101	rd	0110011	R divu
0000001	rs2	rs1	110	rd	0110011	R rem
0000001	rs2	rs1	111	rd	0110011	R remu

Ilustración 2: Estándares de instrucciones tipo R para la multiplicación tiene la estructura de la instrucción, opcodes, tipo de formato y nombres.[1, Fig. 4.2]

Anexo 2: Descripción del circuito VHDL del procesador RISC-V

1.1 Circuito RISC-V

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. USE ieee.std_logic_unsigned.all;
4. use ieee.numeric_std.ALL;
5.
6.
7. entity RiscV is
8.     Port (Reloj : in std_logic;
9.        Reloj2: in STD_LOGIC;
10.       rst: in STD_LOGIC;
11.       num_int: in std_logic;
12.       clk_out, rst_out : out std_logic ;
13.       senales_control: out std_logic_vector(8 down 0);
14.       display_u : out std_logic_vector(6 downto 0);
15.       hab_display : out std_logic_vector(3 downto 0);
16.       dp: out std_logic;
17.       int_display: in std_logic_vector(2 downto 0));
18.
19.
20.end RiscV;
21.
22.architecture Behavioral of RiscV is
23.
24.    component sacar_display is
25.        Port ( valor: in std_logic_vector(31 downto 0);
26.             num_int: in std_logic;
27.             salida1, salida2, salida3,
       salida4: out std_logic_vector(3 downto 0));
28.    end component;
29.
30.    component display is
31.        Port ( Salida : out std_logic_vector(6 downto 0
       );
32.             Entrada : in std_logic_vector(3 downto 0);
33.             dp : out std_logic);
34.    end component;
35.
36.    component Multiplexor9 is
37.        port(  ent1 : in STD_LOGIC_VECTOR (9 downto 0);
38.             ent2 : in STD_LOGIC_VECTOR (9 downto 0);
39.             sel: in std_logic;
40.             Sal : out STD_LOGIC_VECTOR (9 downto 0));
41.    end component;
42.
43.    component Multiplexor32 is
44.        port(  ent1 : in STD_LOGIC_VECTOR (31 downto 0)
       ;
```

```
45.             ent2 : in STD_LOGIC_VECTOR (31 downto 0);
46.             sel: in std_logic;
47.             Sal : out STD_LOGIC_VECTOR (31 downto 0));
48.     end component;
49.
50.     component Sumador_9
51.         port(Ent1 : in STD_LOGIC_vector(9 downto 0);
52.             Ent2 : in STD_LOGIC_vector(9 downto 0);
53.             Sal : out STD_LOGIC_vector(9 downto 0));
54.     end component;
55.
56.     component Contador_programa
57.         Port (  Direcion_in : in STD_LOGIC_VECTOR(9 DOW
        NTO 0);
58.             clk, rst: in STD_LOGIC;
59.             pc : out STD_LOGIC_VECTOR(9 DOWNTO 0));
60.     end component;
61.
62.     component mem_instrucciones
63.         port (
64.            Direcion : in STD_LOGIC_VECTOR(9 DOWNTO 0);
65.            Dato_r : out STD_LOGIC_VECTOR(31 DOWNTO 0));
66.     end component;
67.
68.     component control
69.         Port (instruccion :in std_logic_vector(31 downt
        o 0 );
70.             AluFnt,EscReg,EscMem,MemReg,LeerMem,
        SaltoCond, SaltoIncond, EscDato,
        pc_dato1: out std_logic;
71.             ALUop : out std_logic_vector(3 downto 0 );
72.             ContDatoMem : out STD_LOGIC_VECTOR(2 DOWNT
        O 0));
73.     end component;
74.
75.     component generador_inmediato
76.         Port ( instruccion : in STD_LOGIC_VECTOR (31 do
        wnto 0);
77.             inmediato : out STD_LOGIC_VECTOR (31 downt
        o 0));
78.     end component;
79.
80.     component Registro
81.         Port (     instruccion : in STD_LOGIC_VECTOR(31
         DOWNTO 0);
82.             Dato_s1 : out STD_LOGIC_VECTOR(31 DOWNTO 0
        );
83.             Dato_s2 : out STD_LOGIC_VECTOR(31 DOWNTO 0
        );
84.             Dato_escritura : in STD_LOGIC_VECTOR(31 DO
        WNTO 0);
85.             clk : in STD_LOGIC; -- Escritura sincrona
86.             hw : in STD_LOGIC); -- Habilita escritura
```

```
87.     end component;
88.
89.     component ALU
90.         Port ( instruccion : in STD_LOGIC_VECTOR(31 DOW
       NTO 0);
91.                ALU_control : in STD_LOGIC_VECTOR(4 DOWNTO
        0);
92.                Dato1 : in STD_LOGIC_VECTOR(31 DOWNTO 0);
93.                Dato2 : in STD_LOGIC_VECTOR(31 DOWNTO 0);
94.                zero : out STD_LOGIC;
95.                Resultado : out STD_LOGIC_VECTOR(31 DOWNTO
        0));
96.     end component;
97.
98.     component ALU_control
99.         Port (    instruccion: in std_logic_vector (31
       downto 0);
100.               ALUop: in std_logic_vector (3 downto 0);
101.               ALU: out std_logic_vector (4 downto 0));
102.    end component;
103.
104.    component Memoria_datos is
105.        Port (     Entrada : in STD_LOGIC_VECTOR(11 DO
       WNTO 0);
106.               ContDatoMem : in STD_LOGIC_VECTOR(2 DOWNT
       O 0);
107.               Dato_s : out STD_LOGIC_VECTOR(31 DOWNTO 0
       );
108.               Dato_escritura : in STD_LOGIC_VECTOR(31 D
       OWNTO 0);
109.               hr :in STD_LOGIC; -- Habilita lectura
110.               clk : in STD_LOGIC; -- Escritura sincrona
111.               hw : in STD_LOGIC); -- Habilita escritura
112.    end component;
113.
114.    component Maquina_estados is
115.        Port (clk: in std_logic;
116.              aux: out std_logic_vector(1 downto 0));
117.    end component;
118.
119.    component divisor_frecuencia is
120.        Port (clk: in std_logic;
121.              clk2: out std_logic);
122.    end component;
123.
124.    -- Señales
125.    signal pc_ent, pc_sal, res_sum1, res_inm,
       pc_int : STD_LOGIC_VECTOR(9 DOWNTO 0);
126.    signal instruccion: std_logic_vector(31 downto 0);
127.    --signal addr: std_logic_vector(9 downto 0);
128.    signal inm : std_logic_vector(31 downto 0);
129.    signal dato1, dato2, dato_Alu_1, dato_alu_2,
       dato_esc, res_alu, jal,
```

```
        pc_mux,dato_leidoM,MuxS: std_logic_vector(31 downto
        0);
130.     signal AluF,EscR,EscM,MemR,LeerM, SaltoC, and1,
        saltoInc, EscData, cero, PC_dato1: std_logic;
131.     signal ALUo : std_logic_vector(3 downto 0 );
132.     signal ALUp : std_logic_vector(4 downto 0 );
133.     signal ContDatoMem : std_logic_vector(2 downto 0 )
        ;
134.
135.     signal valor_display : std_logic_vector(3 downto 0
        );
136.     signal valor_int1, valor_int2, valor_int3,
        valor_int4 : std_logic_vector(3 downto 0);
137.     signal aux : std_logic_vector(1 downto 0) :="00";
138.     signal clk2 : std_logic;
139.     signal display_pre : std_logic_vector(31 downto 0)
        ;
140.begin
141.
142.     A1: Contador_programa port map(pc_ent, Reloj, rst,
        pc_sal);
143.
144.     A2: mem_instrucciones port map(pc_sal,instruccion)
        ;
145.     A3: control port map(instruccion,AluF, EscR, EscM,
        MemR,LeerM, SaltoC, saltoInc, EscData, pc_dato1,
        ALUo, ContDatoMem);
146.     A4: generador_inmediato port map(instruccion,
        inm);
147.
148.     A5: Registro port map(instruccion,dato1, dato2,
        dato_esc, reloj,EscR);
149.
150.     A6: ALU_control port map (instruccion, ALUo,
        ALUp);
151.     M1: Multiplexor32 port map(dato2, inm, AluF,
        dato_alu_2);
152.     pc_mux <= "00000000000000000000" & pc_sal;
153.     M2: Multiplexor32 port map(dato1, pc_mux,
        PC_dato1, dato_alu_1);
154.     A7: ALU port map(instruccion, ALUp, dato_alu_1,
        dato_alu_2, cero, res_alu);
155.     A8: Memoria_datos port map( res_alu(11 downto 0),
        ContDatoMem,dato_leidoM, dato2, LeerM, Reloj,EscM);
156.     M3: Multiplexor32 port map(res_alu, dato_leidoM,
        MemR, MuxS);
157.     and1<= SaltoC and cero;
158.     S0: sumador_9 port map("0000000010", pc_sal,
        res_sum1);
159.     S1: sumador_9 port map(inm(9 downto 0), pc_sal,
        res_inm);
160.     M4: Multiplexor9 port map(res_sum1, res_inm,and1,
        pc_int);
```

```
161.     M5: Multiplexor9 port map(pc_int,
       res_alu(9 downto 0), saltoInc, pc_ent);
162.     jal <= ("00000000000000000000000" & res_sum1);
163.     M6: Multiplexor32 port map(MuxS, jal, SaltoInc,
       dato_esc);
164.
165.     --Mostrar por display
166.
167.     display_pre <=  res_alu when int_display = "000" e
       lse
168. "00000000000000000000000" & pc_sal when int_display ="
       001" else
169. dato1 when int_display ="010" else
170. dato2 when int_display ="011"else
171. dato_leidoM when int_display ="100"else
172. inm when int_display ="101"else
173. instruccion when int_display ="110" else
174. dato_alu_2 when int_display ="111";
175.
176.     O1: sacar_display port map(display_pre, num_int,
       valor_int1, valor_int2, valor_int3, valor_int4);
177.     maq0: divisor_frecuencia port map(Reloj2, clk2);
178.     maq1: Maquina_estados port map(clk2, aux);
179.
180.
181.     valor_display <= valor_int1 when aux = "00" else
182. valor_int2 when aux = "01" else
183. valor_int3 when aux = "10" else
184. valor_int4 when aux = "11" else
185. "0000";
186.
187.     hab_display <= "1110" when aux = "00" else
188. "1101" when aux = "01" else
189. "1011" when aux = "10" else
190. "0111" when aux = "11" else
191. "0000";
192.
193.     O2: display port map(display_u, valor_display,
       dp);
194.
195.     -- Mostrar por LEDs
196.     senales_control <= AluF & EscR & EscM & MemR & Lee
       rM & SaltoC & saltoInc & EscData & pc_dato1;
197.     clk_out <= Reloj;
198.     rst_out <= rst;
199.
200.end Behavioral;
```

1.2 Circuito del contador de programa

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4.
5. entity Contador_programa is
6.     Port (  Direcion_in : in STD_LOGIC_VECTOR(9 DOWNTO
   0);
7.             clk, rst: in STD_LOGIC;
8.             pc : out STD_LOGIC_VECTOR(9 DOWNTO 0));
9. end Contador_programa;
10.
11. architecture Behavioral of Contador_programa is
12. signal pc_AUX : STD_LOGIC_VECTOR(9 DOWNTO 0):="000000
   0000";
13. begin
14.     process(clk, rst)
15.     begin
16.         if rst='1' then
17.             pc_AUX <="0000000000";
18.         elsif (clk'event and clk='1') then
19.             pc_AUX <=direcion_in;
20.         end if;
21.     end process;
22.     pc <= pc_AUX;
23. end Behavioral;
```

1.3 Circuito de la memoria de instrucciones.

```
1.    library IEEE;
2.       use IEEE.STD_LOGIC_1164.ALL;
3.       use IEEE.std_logic_unsigned.ALL;
4.
5.
6.       entity mem_instrucciones is
7.
   Port (      Direcion : in STD_LOGIC_VECTOR(9 DOWNTO 0);
8.
     Dato_r : out STD_LOGIC_VECTOR(31 DOWNTO 0));
9.       end mem_instrucciones;
10.
11.architecture Behavioral of mem_instrucciones is
12.     type memoria_type is array (0 to 511) of std_logic
   _vector(31 downto 0);
13.     signal men_inst : memoria_type := (others => X"000
   00000");
14.begin
15.     Dato_r <= men_inst(conv_integer(Direcion(9 downto
   1)));
16.end Behavioral;
```

1.3.1 Instrucciones de la Serie de Fibonacci

```
1. X"00000237", --0
2. X"000A02B7", --2
3. X"00C2D29B", --4
4. X"00000437", --6
5. X"000013B7", --8
6. X"00C3D39B", --10
7. X"00802223", --12
8. X"00702423", --14
9. X"00428763", --16
10.X"000384B3", --18
11.X"008383B3", --20
12.X"00048433", --22
13.X"00722623", --24
14.X"00420213", --26
15.x"01000FE7", --28
```

1.3.2 Instrucciones de la prueba uno

```
1. X"000000B7", --0
2. X"00000963", --2
3. X"02009263", --4
4. X"0220CB63", --6
5. X"04115463", --8
6. X"04116C63", --10
7. x"0600F663", --12
8. X"00000FE7", --14
9. X"00000000", --16
10.X"00000000", --18
11.X"00418133", --20/10
12.x"404100B3",
13.X"00429333",
14.X"005323B3",
15.X"007444B3",
16.X"00625533",
17.x"40625533",
18.X"00A36633",
19.X"00A376B3",
20.X"004000E7",--38
21.X"00f18113",--40
22.x"0901A193",
23.X"0881D113",
24.X"1704E313",
25.X"4AF77293",
26.X"003E1293",
27.x"003E5313",
28.X"403E5393",
29.X"006000E7",
30.X"00000000",
31.X"005021A3",--60
32.x"0070A2A3",
33.X"00812323",
34.X"00B1A3A3",
35.X"00B18423",
36.X"00B194A3",
37.x"008000E7",
38.X"00000000",
39.X"00000000",
40.X"00000000",
41.X"0CD00303",--80
42.x"0CD04383",
43.X"0CF01403",
44.X"0CF05483",
45.X"0D002503",
46.X"00A000E7",
47.x"00000000",
48.X"00000000",
49.X"00000000",
50.X"00000000",
51.X"036A8BB3",--100
```

```
52.x"036A9C33",
53.X"036AACB3",
54.X"036ABD33",
55.x"02C7C833",
56.X"02C959B3",
57.X"02C7E8B3",
58.X"02C97A33",
59.X"00C000E7",
60.X"00000000",
61.X"0A1D0497",--120
62.X"F87FF46F",--122
```

1.4 Circuito del control del procesador

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity control is
5.   Port (instruccion :in std_logic_vector(31 downto 0 )
   ;
6.         AluFnt,EscReg,EscMem,MemReg,LeerMem,
   SaltoCond, SaltoIncond, EscDato,
   PC_dato1: out std_logic;
7.         ALUop : out std_logic_vector(3 downto 0 );
8.         ContDatoMem : out STD_LOGIC_VECTOR(2 DOWNTO 0)
   );
9. end control;
10.
11. architecture Behavioral of control is
12.
13.
14. begin
15.     AluFnt<=instruccion(2)or not(instruccion(6) or in
   struccion(5)) or not(instruccion(6) or instruccion(4))
   ;
16.     EscReg<=not(instruccion(6) or instruccion(5)) or
   instruccion(4);
17.     EscMem<=not(instruccion(6) or instruccion(4)) and
    instruccion(5);
18.     MemReg<=not(instruccion(6) or instruccion(4) or i
   nstruccion(5));
19.     LeerMem<=not(instruccion(6) or instruccion(4) or
   instruccion(5));
20.     SaltoCond<=instruccion(6);
21.     AluOp<=instruccion(6) & instruccion(5) & instrucc
   ion(4) & instruccion(2);
22.     SaltoIncond<=instruccion(6) and instruccion(2);
23.     EscDato<=instruccion(6) and instruccion(2);
24.     PC_dato1 <=instruccion(3) or (not instruccion(5)
   and instruccion(2));
25.     ContDatoMem <=instruccion(14 downto 12);
26. end Behavioral;
```

1.5 Circuito generador de inmediato

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use ieee.numeric_std.ALL;
4. use IEEE.STD_LOGIC_UNSIGNED.ALL;
5.
6.
7. entity generador_inmediato is
8.     Port ( instruccion : in STD_LOGIC_VECTOR (31 downt
   o 0);
9.            inmediato : out STD_LOGIC_VECTOR (31 downto
    0));
10. end generador_inmediato;
11.
12. architecture Behavioral of generador_inmediato is
13.     constant TR:  std_logic_vector(3 downto 0):="0110
   ";
14.     constant TIn:  std_logic_vector(3 downto 0):="001
   0";
15.     constant TL:  std_logic_vector(3 downto 0):="0000
   ";
16.     constant TS:  std_logic_vector(3 downto 0):="0100
   ";
17.     constant TB:  std_logic_vector(3 downto 0):="1100
   ";
18.     constant LUI:  std_logic_vector(3 downto 0):="011
   1";
19.     constant AUIPC:  std_logic_vector(3 downto 0):="0
   011";
20.     constant JAL:  std_logic_vector(3 downto 0):="110
   1";
21. begin
22.
23.     process(instruccion)
24.     variable identificador: STD_LOGIC_VECTOR (3 downt
   o 0);
25.         begin
26.
27.         identificador:= instruccion(6) & instruccion(
   5) & instruccion(4) & instruccion(2);
28.
29.         if((identificador = TL) or (identificador = T
   In))then
30.             if(instruccion(31)='1')then
31.                 inmediato <= "11111111111111111111" &
    instruccion(31 downto 20);
32.             else
33.                 inmediato <= "00000000000000000000" &
    instruccion(31 downto 20);
34.             end if;
35.         elsif(identificador = TS)then
36.             if(instruccion(31)='1')then
```

```
37.                 inmediato <= "11111111111111111111" &
    instruccion(31 downto 25) & instruccion(11 downto 7);
38.             else
39.                 inmediato <= "00000000000000000000" &
    instruccion(31 downto 25) & instruccion(11 downto 7);
40.             end if;
41.         elsif(identificador = TB)then
42.             if(instruccion(31)='1')then
43.                 inmediato <= "11111111111111111111" &
   instruccion(31) & instruccion(7) & instruccion(30 down
   to 25) & instruccion(11 downto 8) & '0';
44.             else
45.                 inmediato <= "00000000000000000000" &
   instruccion(31) & instruccion(7) & instruccion(30 down
   to 25) & instruccion(11 downto 8) & '0';
46.             end if;
47.         elsif(identificador = LUI or identificador =
   AUIPC)then
48.             inmediato <= instruccion(31 downto 12) &
   "000000000000";
49.         elsif(identificador = JAL)then
50.             if(instruccion(3) = '1')then
51.                 if(instruccion(31)='1')then
52.                     inmediato <= "11111111111" & inst
   ruccion(31) & instruccion(19 downto 12) & instruccion(
   20) & instruccion(30 downto 21) & '0';
53.                 else
54.                     inmediato <= "00000000000" & inst
   ruccion(31) & instruccion(19 downto 12) & instruccion(
   20) & instruccion(30 downto 21) & '0';
55.                 end if;
56.             elsif(instruccion(3) = '0')then
57.                 if(instruccion(31)='1')then
58.                     inmediato <= "111111111111111111
   1" & instruccion(31 downto 20);
59.                 else
60.                     inmediato <= "000000000000000000
   0" & instruccion(31 downto 20);
61.                 end if;
62.             end if;
63.
64.         else
65.             inmediato <= "00000000000000000000000000
   00000";
66.         end if;
67.     end process;
68. end Behavioral;
```

1.6 Circuito del Banco de registros

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.std_logic_unsigned.ALL;
4.
5.
6. entity Registro is
7. Port (      instruccion : in STD_LOGIC_VECTOR(31 DOWNTO
    0);
8.            Dato_s1 : out STD_LOGIC_VECTOR(31 DOWNTO 0)
   ;
9.            Dato_s2 : out STD_LOGIC_VECTOR(31 DOWNTO 0)
   ;
10.            Dato_escritura : in STD_LOGIC_VECTOR(31 DO
   WNTO 0);
11.            clk : in STD_LOGIC; -- Escritura sincrona
12.            hw : in STD_LOGIC); -- Habilita escritura
13. end Registro;
14.
15. architecture Behavioral of Registro is
16.     type registros_type is array (31 downto 0) of std
   _logic_vector(31 downto 0);
17.     signal Dir_lectura1 : STD_LOGIC_VECTOR(4 DOWNTO 0
   );
18.     signal Dir_lectura2 :  STD_LOGIC_VECTOR(4 DOWNTO
   0);
19.     signal Dir_escritura :  STD_LOGIC_VECTOR(4 DOWNTO
    0);
20.
21.     signal men_registros : registros_type := ( X"0000
   0010", --31
22.                                             X"0000000A",
23.                                             X"00058001",
24.                                             X"00F0FFFF",
25.                                             X"AAAA0AAA",
26.                                             X"CC10C0CC",
27.                                             x"00001200", -
   -25
28.                                             X"000200A0",
29.                                             X"00000001",
30.                                             X"FF05410F",
31.                                             X"AAA0000A",
32.                                             X"CC10246C", -
   -20
33.                                             x"00001020",
34.                                             X"000000BA",
35.                                             X"0000D010",
36.                                             X"0FEDF00F",
37.                                             X"A0AA543A", -
   -15
38.                                             X"C222222C",
39.                                             x"00130120",
```

```
40.                                             X"0000000A",
41.                                             X"00000001",
42.                                             X"FF784F0F", -
   -10
43.                                             X"0AA45AAA",
44.                                             X"C0C12CCC",
45.                                             x"00040120",
46.                                             X"0000000A",
47.                                             X"00000501", -
   -5
48.                                             X"FF1850FF",
49.                                             X"AAAAAA0A",
50.                                             X"CC165CC0",
51.                                             x"00000000",
52.                                             x"00000000");
53. begin
54.     process(clk, instruccion)
55.     begin
56.         if (clk'event and clk ='1')then
57.             if(hw='1') then
58.                     if(Dir_escritura /= "00000") then
59.                         men_registros(conv_integer(Di
   r_escritura))<= dato_escritura;
60.                     end if;
61.             end if;
62.         end if;
63.     end process;
64.
65.     Dir_lectura1 <= instruccion(19 DOWNTO 15);
66.     Dir_lectura2 <= instruccion(24 DOWNTO 20);
67.     Dir_escritura <= instruccion(11 DOWNTO 7);
68.     Dato_s1 <= men_registros(conv_integer(Dir_lectura
   1));
69.     Dato_s2 <= men_registros(conv_integer(Dir_lectura
   2));
70.
71. end Behavioral;
```

1.7 Circuito del control de la ALU

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity ALU_control is
5.   Port (    instruccion: in std_logic_vector (31 downt
   o 0);
6.             ALUop: in std_logic_vector (3 downto 0);
7.             ALU: out std_logic_vector (4 downto 0));
8. end ALU_control;
9.
10. architecture Behavioral of ALU_control is
11.     signal funct3: std_logic_vector (2 downto 0);
12.     signal opcode: std_logic_vector (6 downto 0);
13.     signal funct7: std_logic_vector (6 downto 0);
14.
15.     constant TR:  std_logic_vector(3 downto 0):="0110
   ";
16.     constant TIn:  std_logic_vector(3 downto 0):="001
   0";
17.     constant TL:  std_logic_vector(3 downto 0):="0000
   ";
18.     constant TS:  std_logic_vector(3 downto 0):="0100
   ";
19.     constant TB:  std_logic_vector(3 downto 0):="1100
   ";
20.     constant LUI:  std_logic_vector(3 downto 0):="011
   1";
21.     constant AUIPC:  std_logic_vector(3 downto 0):="0
   011";
22.     constant JAL:  std_logic_vector(3 downto 0):="110
   1";
23. begin
24.     funct3 <= instruccion(14 downto 12);
25.     opcode <= instruccion(6 downto 0);
26.     funct7 <= instruccion(31 downto 25);
27.     process(instruccion, ALUop, funct7, funct3,
   opcode)
28.     begin
29.         if(ALUop = TR) then
30.             if(funct7(0) = '1') then --multiplicacion
   TM
31.                 ALU <=  (funct3(2) and funct3(1)) &
   (funct3(2) and not(funct3(1))) & ((not(funct3(2)) and
   funct3(0)) or  funct3(1)) & ((not(funct3(2)) and funct
   3(1)) or (funct3(2) and funct3(1)) or (funct3(1) and f
   unct3(0))) & ((funct3(1) and (funct3(2) or funct3(0)))
    or (not(funct3(2)) and (funct3(0) or not(funct3(1))))
    or (not(funct3(1) and funct3(0))));
32.             elsif(funct7(4) = '1') then -- ROR ROL
33.                 ALU <= "100" & funct3(2)& (funct3(2)
   and funct3(0));
```

```
34.             else -- originales R
35.                 ALU <= funct3 & funct7(5) & funct7(0)
    ;
36.             end if;
37.         elsif(ALUop = TIn) then -- tipo I
38.             ALU <= funct3 & "00";
39.         elsif(ALUop = TB) then -- Tipo B
40.             ALU <= funct3(0) & (funct3(2) or  funct3(
    1)) & (funct3(2) and funct3(1)) & not(funct3(2)) & '1'
    ;
41.         elsif(ALUop = TS or ALUop = TL) then -- Tipo
    L o S
42.             ALU <= "00000";
43.         elsif(ALUop = LUI) then --LUI
44.             ALU <= "11111";
45.         elsif(ALUop = LUI) then
46.             ALU <= "00000";
47.         elsif(ALUop = JAL) then --Tipo J
48.             if(instruccion(3)='1') then  --JALR
49.                    ALU <= "11011";
50.             elsif(instruccion(3)='0') then --JAL
51.                    ALU <= "11010";
52.             end if;
53.         end if;
54.
55.     end process;
56.
57. end Behavioral;
```

1.8 Circuito de la ALU

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. USE ieee.std_logic_unsigned.all;
4. use ieee.numeric_std.ALL;
5.
6. entity ALU is
7.     Port ( instruccion : in STD_LOGIC_VECTOR(31 DOWNTO
    0);
8.           ALU_control : in STD_LOGIC_VECTOR(4 DOWNTO 0)
   ;
9.           Dato1 : in STD_LOGIC_VECTOR(31 DOWNTO 0);
10.          Dato2 : in STD_LOGIC_VECTOR(31 DOWNTO 0);
11.          zero : out STD_LOGIC;
12.          Resultado : out STD_LOGIC_VECTOR(31 DOWNTO 0)
   );
13. end ALU;
14.
15. architecture Behavioral of ALU is
16.     --Operaciones matematicas
17.     constant Suma : std_logic_vector(4 downto 0):="00
   000";-- Suma
18.     constant Resta : std_logic_vector(4 downto 0):="0
   0010";-- Resta
19.     --Operaciones logicas
20.     constant op_and : std_logic_vector(4 downto 0):="
   11100";-- Operacion AND
21.     constant op_or : std_logic_vector(4 downto 0):="1
   1000";-- Operacion OR
22.     constant op_XOR : std_logic_vector(4 downto 0):="
   10000";-- Operacion XOR
23.     --Desplazamientos y rotaciones
24.     constant op_SRA : std_logic_vector(4 downto 0):="
   10110";-- desplazamiento aritmetico a la derecha
25.     constant op_SRL : std_logic_vector(4 downto 0):="
   10100";-- desplazamiento logico a la derecha
26.     constant op_SLL : std_logic_vector(4 downto 0):="
   00100";-- desplazamiento logico a la izquierda
27.     constant op_ROL : std_logic_vector(4 downto 0):="
   10001";-- Rotacion a la izquierda
28.     constant op_ROR : std_logic_vector(4 downto 0):="
   10010";-- Rotacion a la derecha
29.     --Comparaciones
30.     constant op_SLTU : std_logic_vector(4 downto 0):=
   "01100";-- A<B  escribe en el destino 1 si se cumple
   la condicion, 0 si no lo cumple.(sin signo)
31.     constant op_SLT : std_logic_vector(4 downto 0):="
   01000";-- escribe en el destino 1 si se cumple la
   condicion, 0 si no lo cumple.(con signo)
32.     --Multiplicacion y division
33.
34.     --Comparaciones Salida Salt cond
```

```
35.     constant op_BLT : std_logic_vector(4 downto 0):="
   01011";--A>=B
36.     constant op_BGEU : std_logic_vector(4 downto 0):=
   "11101";--
37.     constant op_BGE : std_logic_vector(4 downto 0):="
   11001";--
38.     constant op_BLTU : std_logic_vector(4 downto 0):=
   "01101";--
39.     constant op_Bne : std_logic_vector(4 downto 0):="
   10011";--A/=B
40.     constant op_Beq : std_logic_vector(4 downto 0):="
   00011";--A=B
41.
42.     --JAL y JALR
43.     constant op_JAL : std_logic_vector(4 downto 0):="
   11010";-- JAL
44.     constant op_JALR : std_logic_vector(4 downto 0):=
   "11011";-- JALR
45.
46.     -- Funciones de multiplicacion
47.     constant op_MUL : std_logic_vector(4 downto 0):="
   00001";-- MultiplicaciÃ³n parte baja
48.     constant op_Mulh : std_logic_vector(4 downto 0):=
   "00101";-- MultiplicaciÃ³n parte alta signed
49.     constant op_Mulhu : std_logic_vector(4 downto 0):
   ="00111";-- MultiplicaciÃ³n parte alta unsigned
50.     constant op_Mulhsu : std_logic_vector(4 downto 0)
   :="00110";-- MultiplicaciÃ³n parte alta
   signed*unsigned
51.     constant op_div : std_logic_vector(4 downto 0):="
   01001";-- DivisiÃ³n signed
52.     constant op_divu : std_logic_vector(4 downto 0):=
   "01010";-- DivisiÃ³n unsigned
53.     constant op_rem : std_logic_vector(4 downto 0):="
   01110";-- Resto signed
54.     constant op_remu : std_logic_vector(4 downto 0):=
   "01111";-- Resto unsigned
55.
56.     -- Funciones especiales
57.     constant op_LUI : std_logic_vector(4 downto 0):="
   11111";
58.
59.
60. begin
61.     process(Dato1, Dato2, ALU_control,instruccion)
62.         variable Result: STD_LOGIC_VECTOR(31 DOWNTO 0
   );
63.         variable Res_mul: STD_LOGIC_VECTOR(63 DOWNTO
   0);
64.     begin
65.
66.         if(ALU_control=suma) then
67.             Result := Dato1 + Dato2;
```

```
68.
69.             if(Result="00000000000000000000000000000
   00") then
70.                 zero <= '0';
71.             else
72.                 zero <= '1';
73.             end if;
74.
75.         elsif (ALU_control=resta) then
76.             Result:= Dato1 - Dato2;
77.             if(Result="00000000000000000000000000000
   00") then
78.                 zero <= '0';
79.             else
80.                 zero <= '1';
81.             end if;
82.
83.         elsif (ALU_control=op_and) then
84.             Result:= Dato1 and Dato2;
85.             if(Result="00000000000000000000000000000
   00") then
86.                 zero <= '0';
87.             else
88.                 zero <= '1';
89.             end if;
90.
91.         elsif (ALU_control=op_or) then
92.             Result:= Dato1 or Dato2;
93.             if(Result="00000000000000000000000000000
   00") then
94.                 zero <= '0';
95.             else
96.                 zero <= '1';
97.             end if;
98.
99.         elsif (ALU_control=op_xor) then
100.             Result:= Dato1 xor Dato2;
101.             if(Result="00000000000000000000000000000
   000") then
102.                 zero <= '0';
103.             else
104.                 zero <= '1';
105.             end if;
106.
107.         elsif (ALU_control = op_SLT) then
108.             if( signed(Dato1) < signed(Dato2) ) then
109.                 Result:= "00000000000000000000000000
   000001";
110.                 zero <= '1';
111.             else
112.                 Result:= "00000000000000000000000000
   000000";
113.                 zero <= '0';
```

```
114.             end if;
115.
116.         elsif (ALU_control = op_SLTU) then
117.             if( unsigned(Dato1) < unsigned(Dato2) )
   then
118.                 Result:= "00000000000000000000000000
   000001";
119.                 zero <= '1';
120.             else
121.                 Result:= "00000000000000000000000000
   000000";
122.                 zero <= '0';
123.             end if;
124.
125.         elsif (ALU_control=op_SRA) then
126.             Result:= std_logic_vector(shift_right(si
   gned(dato1), to_integer(unsigned(dato2))));
127.             if(Result="00000000000000000000000000000
   000") then
128.                 zero <= '0';
129.             else
130.                 zero <= '1';
131.             end if;
132.
133.         elsif (ALU_control=op_SRL) then
134.             Result:= std_logic_vector(shift_right(un
   signed(dato1), to_integer(unsigned(dato2))));
135.             if(Result="00000000000000000000000000000
   000") then
136.                 zero <= '0';
137.             else
138.                 zero <= '1';
139.             end if;
140.
141.         elsif (ALU_control=op_SLL) then
142.             Result:= std_logic_vector(shift_left(uns
   igned(dato1), to_integer(unsigned(dato2))));
143.             if(Result="00000000000000000000000000000
   000") then
144.                 zero <= '0';
145.             else
146.                 zero <= '1';
147.             end if;
148.
149.         elsif (ALU_control=op_ROR) then
150.             Result:= std_logic_vector(rotate_right(u
   nsigned(dato1), to_integer(unsigned(dato2))));
151.             if(Result="00000000000000000000000000000
   000") then
152.                 zero <= '0';
153.             else
154.                 zero <= '1';
155.             end if;
```

```
156.
157.         elsif (ALU_control=op_ROL) then
158.             Result:= std_logic_vector(rotate_left(un
   signed(dato1), to_integer(unsigned(dato2))));
159.             if(Result="00000000000000000000000000000
   000") then
160.                 zero <= '0';
161.             else
162.                 zero <= '1';
163.             end if;
164.         --JALR
165.         elsif (ALU_control=op_JALR) then
166.             Result:= dato1+ dato2;
167.             if(Result="00000000000000000000000000000
   000") then
168.                 zero <= '0';
169.             else
170.                 zero <= '1';
171.             end if;
172.         --JAL
173.         elsif (ALU_control=op_JAL) then
174.             Result:= dato1 + dato2;
175.             if(Result="00000000000000000000000000000
   000") then
176.                 zero <= '0';
177.             else
178.                 zero <= '1';
179.             end if;
180.
181.         -- Tipo M
182.         elsif (ALU_control=op_Mul) then
183.             Res_Mul:= dato1 * dato2;
184.             Result:=Res_Mul(31 downto 0);
185.             if(Result="00000000000000000000000000000
   000") then
186.                 zero <= '0';
187.             else
188.                 zero <= '1';
189.             end if;
190.
191.         elsif (ALU_control=op_Mulh) then
192.             Res_Mul:=std_logic_vector( signed(dato1)
    * signed(dato2));
193.             Result:=Res_Mul(63 downto 32);
194.             if(Result="00000000000000000000000000000
   000") then
195.                 zero <= '0';
196.             else
197.                 zero <= '1';
198.             end if;
199.
200.         elsif (ALU_control=op_Mulhu) then
```

```
201.             Res_Mul:= std_logic_vector(unsigned(dato
   1) * unsigned(dato2));
202.             Result:=Res_Mul(63 downto 32);
203.             if(Result="00000000000000000000000000000
   000") then
204.                 zero <= '0';
205.             else
206.                 zero <= '1';
207.             end if;
208.
209.         elsif (ALU_control=op_Mulhsu) then
210.             Res_Mul:= std_logic_vector(to_signed(TO_
   INTEGER(signed(dato1)) * TO_INTEGER(unsigned(dato2)),R
   es_Mul'length));
211.             Result:=Res_Mul(63 downto 32);
212.             if(Result="00000000000000000000000000000
   000") then
213.                 zero <= '0';
214.             else
215.                 zero <= '1';
216.             end if;
217.
218.         elsif (ALU_control=op_divu) then
219.             if Dato1 /= x"00000000" and Dato2 /= x"0
   0000000" then
220.                 Result:= std_logic_vector(unsigned(d
   ato1) / unsigned(dato2));
221.                 if(Result="00000000000000000000000000
   0000000") then
222.                     zero <= '0';
223.                 else
224.                     zero <= '1';
225.                 end if;
226.             end if;
227.
228.         elsif (ALU_control=op_div) then
229.             if Dato1 /= x"00000000" and Dato2 /= x"0
   0000000" then
230.                 Result:= std_logic_vector(signed(dat
   o1) / signed(dato2));
231.                 if(Result="00000000000000000000000000
   0000000") then
232.                     zero <= '0';
233.                 else
234.                     zero <= '1';
235.                 end if;
236.             end if;
237.         elsif (ALU_control=op_rem) then
238.             if Dato1 /= x"00000000" and Dato2 /= x"0
   0000000" then
239.                 Result:= std_logic_vector(signed(dat
   o1) rem signed(dato2));
```

```
240.                 if(Result="00000000000000000000000000
   0000000") then
241.                     zero <= '0';
242.                 else
243.                     zero <= '1';
244.                 end if;
245.             end if;
246.         elsif (ALU_control=op_remu) then
247.             if Dato1 /= x"00000000" and Dato2 /= x"0
   0000000" then
248.                 Result:= std_logic_vector(unsigned(d
   ato1) rem unsigned(dato2));
249.                 if(Result="00000000000000000000000000
   0000000") then
250.                     zero <= '0';
251.                 else
252.                     zero <= '1';
253.                 end if;
254.             end if;
255.
256.         elsif(ALU_control=op_LUI) then
257.             Result:= Dato2;
258.             if(Result="00000000000000000000000000000
   000") then
259.                 zero <= '0';
260.             else
261.                 zero <= '1';
262.             end if;
263.
264.         elsif (ALU_control=op_BGEU) then
265.             if(unsigned(Dato1) >= unsigned(Dato2)) t
   hen
266.                 zero <= '1';
267.             else
268.                 zero <= '0';
269.             end if;
270.         elsif (ALU_control=op_BGE) then
271.             if( signed(Dato1) >= signed(Dato2) ) the
   n
272.                 zero <= '1';
273.             else
274.                 zero <= '0';
275.             end if;
276.
277.         elsif (ALU_control=op_BNE) then
278.             if( signed(Dato1) /= signed(Dato2) ) the
   n
279.                 zero <= '1';
280.             else
281.                 zero <= '0';
282.             end if;
283.         elsif (ALU_control=op_BEQ) then
284.             if( signed(Dato1) = signed(Dato2) ) then
```

```
285.                 zero <= '1';
286.             else
287.                 zero <= '0';
288.             end if;
289.
290.         elsif (ALU_control=op_BLT) then
291.             if( signed(Dato1) < signed(Dato2) ) then
292.                 zero <= '1';
293.             else
294.                 zero <= '0';
295.             end if;
296.
297.         elsif (ALU_control=op_BLTU) then
298.             if( unsigned(Dato1) < unsigned(Dato2) )
  then
299.                 zero <= '1';
300.             else
301.                 zero <= '0';
302.             end if;
303.
304.         end if;
305.
306.         Resultado<=result;
307.     end process;
308.
309. end Behavioral;
```

1.9 Circuito de la memoria de datos

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.std_logic_unsigned.ALL;
4.
5. entity Memoria_datos is
6. Port (      Entrada : in STD_LOGIC_VECTOR(11 DOWNTO 0);
7.             ContDatoMem : in STD_LOGIC_VECTOR(2 DOWNTO 0
   );
8.             Dato_s : out STD_LOGIC_VECTOR(31 DOWNTO 0);
9.             Dato_escritura : in STD_LOGIC_VECTOR(31 DOWN
   TO 0);
10.             hr :in STD_LOGIC; -- Habilita lectura
11.             clk : in STD_LOGIC; -- Escritura sincrona
12.             hw : in STD_LOGIC); -- Habilita escritura
13. end Memoria_datos;
14.
15. architecture Behavioral of Memoria_datos is
16.
17.     type memoria_type is array (0 to 1024) of std_logi
   c_vector(31 downto 0);
18.     signal men_datos : memoria_type:= (X"000283B3",
19.                                        X"00780A13",
20.                                        X"415A52B3",
21.                                        X"00600093",
22.                                        X"01F30D33",
23.                                        X"FE078BE3",
24.                                        x"000033EF",
25.                                        others => X"907
   80A13");-- Datos de la memoria
26.     signal Direcion : std_logic_vector (9 downto 0);
27.     signal posicion : std_logic_vector (1 downto 0);
28.
29. begin
30.     Direcion <= Entrada(11 downto 2);
31.     posicion <= Entrada(1 downto 0);
32.
33.     process(clk)
34.     begin
35.     if clk'event and clk ='1' then
36.         if hw='1' then
37.             if ContDatoMem = "000" then
38.                 if posicion="00" then
39.                     men_datos(conv_integer(Direcion))(
   7 downto 0)<= Dato_escritura(7 downto 0);
40.                 elsif posicion="01" then
41.                     men_datos(conv_integer(Direcion))(
   15 downto 8)<= Dato_escritura(7 downto 0);
42.                 elsif posicion="10" then
43.                     men_datos(conv_integer(Direcion))(
   23 downto 16)<= Dato_escritura(7 downto 0);
44.                 else
```

```
45.                     men_datos(conv_integer(Direcion))(
   31 downto 24)<=  Dato_escritura(7 downto 0);
46.                 end if;
47.
48.             elsif ContDatoMem = "001" then
49.                 if posicion="00" or posicion="01" then
50.                     men_datos(conv_integer(Direcion))(
   15 downto 0)<= Dato_escritura(15 downto 0);
51.                 elsif posicion="10" or posicion="11" t
   hen
52.                     men_datos(conv_integer(Direcion))(
   31 downto 16)<= Dato_escritura(15 downto 0);
53.                 end if;
54.             elsif ContDatoMem = "010" then
55.                 men_datos(conv_integer(Direcion))<= Da
   to_escritura;
56.             end if;
57.         end if;
58.     end if;
59.     end process;
60.
61.
62.     process(Entrada, dato_escritura, hr, clk,
   ContDatoMem, Direcion)
63.     variable Dato_i1 : std_logic_vector (31 downto 0);
64.     variable f8 : std_logic_vector (7 downto 0);
65.     variable f16 : std_logic_vector (15 downto 0);
66.     begin
67.         if hr='1' then
68.             Dato_i1 := men_datos(conv_integer(Direcion
   ));
69.
70.             if ContDatoMem = "000" or ContDatoMem = "1
   00" then
71.                 if posicion="00" then
72.                     f8 := Dato_i1(7 downto 0);
73.                 elsif posicion="01" then
74.                     f8 := Dato_i1(15 downto 8);
75.                 elsif posicion="10" then
76.                     f8 := Dato_i1(23 downto 16);
77.                 elsif posicion="11" then
78.                     f8 := Dato_i1(31 downto 24);
79.                 end if;
80.
81.                 if ContDatoMem = "100" or (f8(7)='0' a
   nd ContDatoMem = "000")  then
82.                   Dato_s <= "00000000000000000000000000"
    & f8;
83.                 elsif f8(7)='1' and ContDatoMem = "000
   " then
84.                   Dato_s <= "11111111111111111111111111"
    & f8;
85.                 end if;
```

```
86.
87.             elsif ContDatoMem = "001" or ContDatoMem =
    "101" then
88.                 if posicion="00" or posicion="01" then
89.                     f16 := Dato_i1(15 downto 0);
90.                 elsif posicion="10" or posicion="11" t
   hen
91.                     f16 := Dato_i1(31 downto 16);
92.                 end if;
93.                 if ContDatoMem = "101" or (f16(15)='0'
    and ContDatoMem = "001")  then
94.                   Dato_s <= "0000000000000000" & f16;

95.                 elsif f16(15)='1' and ContDatoMem = "0
   01" then
96.                   Dato_s <= "1111111111111111" & f16;
97.                 end if;
98.
99.             elsif ContDatoMem = "010" then
100.                 Dato_s <= Dato_i1;
101.
102.             end if;
103.
104.         end if;
105.     end process;
106. end Behavioral;
```

1.10 Circuito multiplexor

1.10.1 Multiplexor 32 bits

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity Multiplexor32 is
5.     Port(ent1 : in STD_LOGIC_VECTOR (31 downto 0);
6.          ent2 : in STD_LOGIC_VECTOR (31 downto 0);
7.          sel: in std_logic;
8.          Sal : out STD_LOGIC_VECTOR (31 downto 0));
9. end Multiplexor32;
10.
11. architecture Behavioral of Multiplexor32 is
12. begin
13.     Sal <= ent1 when sel='0' else
14.  ent2 when sel='1';
15. end Behavioral;
```

1.10.2 Multiplexor 10 bits

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity Multiplexor9 is
5.     Port(ent1 : in STD_LOGIC_VECTOR (9 downto 0);
6.          ent2 : in STD_LOGIC_VECTOR (9 downto 0);
7.          sel: in std_logic;
8.          Sal : out STD_LOGIC_VECTOR (9 downto 0));
9. end Multiplexor9;
10.
11. architecture Behavioral of Multiplexor9 is
12. begin
13.     Sal <= ent1 when sel='0' else
14.  ent2 when sel='1';
15. end Behavioral;
```

1.11 Circuito sumador

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.numeric_std.ALL;
4.
5. entity Sumador_9 is
6.     Port(Ent1 : in STD_LOGIC_vector(9 downto 0);
7.          Ent2 : in STD_LOGIC_vector(9 downto 0);
8.          Sal : out STD_LOGIC_vector(9 downto 0));
9. end Sumador_9;
10.
11.architecture Behavioral of Sumador_9 is
12.begin
13.    Sal <= std_logic_vector (unsigned(Ent1) + unsigne
   d(Ent2));
14.end Behavioral;
```

1.12 Circuito divisor de frecuencia

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity divisor_frecuencia is
5.     Port (clk : in std_logic;
6.           clk2 : out std_logic);
7. end divisor_frecuencia;
8.
9. architecture Behavioral of divisor_frecuencia is
10.     constant max_count: INTEGER := 5000;
11.     signal count: INTEGER range 0 to max_count;
12.     signal clk_state: STD_LOGIC := '0';
13. begin
14.     gen_clock: process(clk, clk_state, count)
15.     begin
16.         if clk'event and clk='1' then
17.             if count < max_count then
18.                 count <= count+1;
19.             else
20.                 clk_state <= not clk_state;
21.                 count <= 0;
22.             end if;
23.         end if;
24.     end process;
25.     persecond: process (clk_state)
26.     begin
27.         clk2 <= clk_state;
28.     end process;
29. end Behavioral;
```

1.13 Circuito máquina de estados

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity Maquina_estados is
5.     Port (clk: in std_logic;
6.           aux: out std_logic_vector(1 downto 0));
7. end Maquina_estados;
8.
9. architecture Behavioral of Maquina_estados is
10.     type STATES is (S0, S1, S2, S3);
11.     signal state_reg, state_next: STATES;
12. begin
13.     process(CLK)
14.     begin
15.         if CLK'event and CLK='1' then
16.             state_reg <= state_next;
17.         end if;
18.     end process;
19.     -- Lógica de estado siguiente (circuito
   combinacional)
20.     process (state_reg)
21.     begin
22.         state_next <= state_reg;
23.         case state_reg is
24.             when S0 =>
25.                 aux <="00";
26.                 state_next <= S1;
27.             when S1 =>
28.                 aux <="01";
29.                 state_next <= S2;
30.             when S2 =>
31.                 aux <="10";
32.                 state_next <= S3;
33.             when S3 =>
34.                 aux <="11";
35.                 state_next <= S0;
36.         end case;
37.     end process;
38. end Behavioral;
```

1.14 Circuito sacar display

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. USE ieee.std_logic_unsigned.all;
4. use ieee.numeric_std.ALL;
5. use ieee.Std_Logic_Arith;
6.
7. entity sacar_display is
8.   Port ( valor: in std_logic_vector(31 downto 0);
9.          num_int: in std_logic;
10.          salida1, salida2, salida3,
    salida4: out std_logic_vector(3 downto 0));
11. end sacar_display;
12.
13. architecture Behavioral of sacar_display is
14. begin
15.  process(valor, num_int)
16.  begin
17.     if num_int ='0' then
18.         salida1 <= valor(3 downto 0);
19.         salida2 <= valor(7 downto 4);
20.         salida3 <= valor(11 downto 8);
21.         salida4 <= valor(15 downto 12);
22.     elsif num_int ='1' then
23.         salida1 <= valor(19 downto 16);
24.         salida2 <= valor(23 downto 20);
25.         salida3 <= valor(27 downto 24);
26.         salida4 <= valor(31 downto 28);
27.     end if;
28.  end process;
29. end Behavioral;
```

1.15 Circuito display

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity display is
5.     Port ( Salida : out std_logic_vector(6 downto 0);
6.          Entrada : in std_logic_vector(3 downto 0);
7.          dp : out std_logic);
8. end display;
9.
10. architecture Behavioral of display is
11. begin
12.    Salida <="1000000" when Entrada = "0000" else -- 0
13.             "1111001" when Entrada = "0001" else -- 1
14.             "0100100" when Entrada = "0010" else -- 2
15.             "0110000" when Entrada = "0011" else -- 3
16.             "0011001" when Entrada = "0100" else -- 4
17.             "0010010" when Entrada = "0101" else -- 5
18.             "0000010" when Entrada = "0110" else -- 6
19.             "1111000" when Entrada = "0111" else -- 7
20.             "0000000" when Entrada = "1000" else -- 8
21.             "0010000" when Entrada = "1001" else -- 9
22.             "0001000" when Entrada = "1010" else -- A
23.             "0000000" when Entrada = "1011" else -- B
24.             "1000110" when Entrada = "1100" else -- C
25.             "1000000" when Entrada = "1101" else -- D
26.             "0000110" when Entrada = "1110" else -- E
27.             "0001110" when Entrada = "1111" else -- F
28.             "1111111";
29.     dp <= '0' when Entrada = "1011" else -- B
30.           '0' when Entrada = "1101" else -- D
31.           '1';
32. end Behavioral;
```

1.16 Constraint

```
1. # Clock signal
2. set_property PACKAGE_PIN W5 [get_ports {Reloj2}]
3. set_property IOSTANDARD LVCMOS33 [get_ports {Reloj2}]
4. create_clock -period 100000.000 -name Reloj2 -
   waveform {0.000 50000.000} [get_ports Reloj2]
5.
6. # Switches
7. set_property PACKAGE_PIN V17 [get_ports {num_int}]
8. set_property IOSTANDARD LVCMOS33 [get_ports {num_int}]
9. set_property PACKAGE_PIN
   V16 [get_ports {int_display[0]}]
10. set_property IOSTANDARD
   LVCMOS33 [get_ports {int_display[0]}]
11. set_property PACKAGE_PIN
   W16 [get_ports {int_display[1]}]
12. set_property IOSTANDARD
   LVCMOS33 [get_ports {int_display[1]}]
13. set_property PACKAGE_PIN
   W17 [get_ports {int_display[2]}]
14. set_property IOSTANDARD
   LVCMOS33 [get_ports {int_display[2]}]
15.
16. set_property PACKAGE_PIN T1 [get_ports Reloj]
17. set_property IOSTANDARD LVCMOS33 [get_ports Reloj]
18. set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets
   clk_out]
19. set_property PACKAGE_PIN R2 [get_ports rst]
20. set_property IOSTANDARD LVCMOS33 [get_ports rst]
21.
22.
23. # LEDs
24. set_property PACKAGE_PIN
   U16 [get_ports {senales_control[0]}]
25. set_property IOSTANDARD
   LVCMOS33 [get_ports {senales_control[0]}]
26. set_property PACKAGE_PIN
   E19 [get_ports {senales_control[1]}]
27. set_property IOSTANDARD
   LVCMOS33 [get_ports {senales_control[1]}]
28. set_property PACKAGE_PIN
   U19 [get_ports {senales_control[2]}]
29. set_property IOSTANDARD
   LVCMOS33 [get_ports {senales_control[2]}]
30. set_property PACKAGE_PIN
   V19 [get_ports {senales_control[3]}]
31. set_property IOSTANDARD
   LVCMOS33 [get_ports {senales_control[3]}]
32. set_property PACKAGE_PIN
   W18 [get_ports {senales_control[4]}]
33. set_property IOSTANDARD
   LVCMOS33 [get_ports {senales_control[4]}]
```

```
set_property PACKAGE_PIN
U15 [get_ports {senales_control[5]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {senales_control[5]}]
set_property PACKAGE_PIN
U14 [get_ports {senales_control[6]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {senales_control[6]}]
set_property PACKAGE_PIN
V14 [get_ports {senales_control[7]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {senales_control[7]}]
set_property PACKAGE_PIN
V13 [get_ports {senales_control[8]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {senales_control[8]}]

set_property PACKAGE_PIN P1 [get_ports clk_out]
set_property IOSTANDARD LVCMOS33 [get_ports clk_out]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets
clk_out_OBUF]
set_property PACKAGE_PIN L1 [get_ports rst_out]
set_property IOSTANDARD LVCMOS33 [get_ports rst_out]

#7 segment display
set_property PACKAGE_PIN
W7 [get_ports {display_u[0]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {display_u[0]}]
set_property PACKAGE_PIN
W6 [get_ports {display_u[1]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {display_u[1]}]
set_property PACKAGE_PIN
U8 [get_ports {display_u[2]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {display_u[2]}]
set_property PACKAGE_PIN
V8 [get_ports {display_u[3]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {display_u[3]}]
set_property PACKAGE_PIN
U5 [get_ports {display_u[4]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {display_u[4]}]
set_property PACKAGE_PIN
V5 [get_ports {display_u[5]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {display_u[5]}]
set_property PACKAGE_PIN
U7 [get_ports {display_u[6]}]
set_property IOSTANDARD
LVCMOS33 [get_ports {display_u[6]}]
```

```
64.
65. set_property PACKAGE_PIN V7 [get_ports dp]
66. set_property IOSTANDARD LVCMOS33 [get_ports dp]
67.
68. set_property PACKAGE_PIN
    U2 [get_ports {hab_display[0]}]
69. set_property IOSTANDARD
    LVCMOS33 [get_ports {hab_display[0]}]
70. set_property PACKAGE_PIN
    U4 [get_ports {hab_display[1]}]
71. set_property IOSTANDARD
    LVCMOS33 [get_ports {hab_display[1]}]
72. set_property PACKAGE_PIN
    V4 [get_ports {hab_display[2]}]
73. set_property IOSTANDARD
    LVCMOS33 [get_ports {hab_display[2]}]
74. set_property PACKAGE_PIN
    W4 [get_ports {hab_display[3]}]
75. set_property IOSTANDARD
    LVCMOS33 [get_ports {hab_display[3]}]
```

Anexo 3: Descripción de los circuitos multiplicadores en VHDL.

1.1 Circuito del Multiplicador Paralelo CSA.

```
1. ----------------------------------------------------
2. -- parallel_csa_multiplier.vhd
3. --
4. -- section 8.2.2 parallel carry save adder(CSA)
   multiplier
5. --
6. -- Computes: z = x·y
7. -- x: n bits
8. -- y: m bits
9. -- z: n+m bits
10. -- for n greater than or equal to m
11. --
12. ----------------------------------------------------
13.
14. LIBRARY IEEE;
15. USE IEEE.STD_LOGIC_1164.ALL;
16. USE IEEE.STD_LOGIC_ARITH.ALL;
17. USE IEEE.STD_LOGIC_UNSIGNED.ALL;
18. ENTITY parallel_csa_multiplier IS
19.   GENERIC(n,m: NATURAL:= 128);
20. PORT(
21.   x: IN STD_LOGIC_VECTOR(n-1 DOWNTO 0);
22.   y: IN STD_LOGIC_VECTOR(m-1 DOWNTO 0);
23.   z: OUT STD_LOGIC_VECTOR(n+m-1 DOWNTO 0)
24. );
25. END parallel_csa_multiplier;
26.
27. ARCHITECTURE circuit OF parallel_csa_multiplier IS
28.   TYPE matrix IS ARRAY (0 TO m-
   1) OF STD_LOGIC_VECTOR(n-1 DOWNTO 0);
29.   SIGNAL c, d, e, f: matrix;
30.   SIGNAL first_operand: STD_LOGIC_VECTOR(n-
   2 DOWNTO 0);
31.   SIGNAL second_operand: STD_LOGIC_VECTOR(n-
   1 DOWNTO 0);
32. BEGIN
33.   main_iteration: FOR i IN 0 TO m-1 GENERATE
34.     internal_iteration: FOR j IN 0 TO n-1 GENERATE
35.       f(i)(j) <= (x(j) AND y(i)) XOR c(i)(j) XOR d(i)
   (j);
36.       e(i)(j) <= (x(j) AND y(i) AND c(i)(j)) OR (x(j)
    AND y(i) AND d(i)(j)) OR (c(i)(j) AND d(i)(j));
37.     END GENERATE;
38.   END GENERATE;
39.
40.   connections1: FOR j IN 0 TO n-1 GENERATE
41.     c(0)(j) <= '0';
```

```
42.   END GENERATE;
43.   connections2: FOR i IN 1 TO m-1 GENERATE
44.     connections3: FOR j IN 0 TO n-2 GENERATE
45.       c(i)(j) <= f(i-1)(j+1);
46.     END GENERATE;
47.     c(i)(n-1) <= '0';
48.   END GENERATE;
49.
50.   connections4: FOR j IN 0 TO m-1 GENERATE
51.     d(0)(j) <= '0';
52.   END GENERATE;
53.   connections5: FOR j IN m TO n-1 GENERATE
54.     d(0)(j) <= '0';
55.   END GENERATE;
56.   connections6: FOR i IN 1 TO m-1 GENERATE
57.     connections7: FOR j IN 0 TO n-1 GENERATE
58.       d(i)(j) <= e(i-1)(j);
59.     END GENERATE;
60.   END GENERATE;
61.
62.   outputs: FOR j IN 0 TO m-1 GENERATE
63.     z(j) <= f(j)(0);
64.   END GENERATE;
65.   first_operand<= f(m-1)(n-1 DOWNTO 1);
66.   second_operand <= e(m-1);
67.   z(n+m-
   1 DOWNTO m) <= first_operand + second_operand;
68.
69. END circuit;
```

1.2 Circuito Multiplicador sintetizado por Xilinx

```
1.  LIBRARY IEEE;
2.  USE IEEE.STD_LOGIC_1164.ALL;
3.  USE IEEE.STD_LOGIC_ARITH.ALL;
4.  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5.
6.  ENTITY multiplier IS
7.    GENERIC(n: NATURAL:= 128);
8.  PORT(
9.    x: IN STD_LOGIC_VECTOR(n-1 DOWNTO 0);
10.   y: IN STD_LOGIC_VECTOR(n-1 DOWNTO 0);
11.   z: OUT STD_LOGIC_VECTOR(n+n-1 DOWNTO 0)
12. );
13. END multiplier;
14.
15. ARCHITECTURE circuit OF multiplier IS
16.
17. BEGIN
18.   z <= x*y;
19. END circuit;
```

1.3 Circuito Multiplicador de Árbol de Wallace

1.3.1 4 bits

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity Multiplicador is
5. Port (  DatoA, DatoB : in STD_LOGIC_VECTOR(3 DOWNTO 0);
6.     Resultado : out STD_LOGIC_VECTOR(7 DOWNTO 0));
7. end Multiplicador;
8.
9. architecture Behavioral of Multiplicador is
10.  component F_adder is
11.  port(  res : out STD_LOGIC;
12.      carry : out STD_LOGIC;
13.      a : in STD_LOGIC;
14.      b : in STD_LOGIC;
15.      c : in STD_LOGIC);
16.  end component;
17.  component H_adder is
18.  port(  res : out STD_LOGIC;
19.      carry : out STD_LOGIC;
20.      a : in STD_LOGIC;
21.      b : in STD_LOGIC);
22.  end component;
23.
24.    signal a0_0_0 : std_logic;
25.    signal a0_1_0 : std_logic;
26.    signal a0_2_0 : std_logic;
27.    signal a0_3_0 : std_logic;
28.    signal a0_1_1 : std_logic;
29.    signal a0_2_1 : std_logic;
30.    signal a0_3_1 : std_logic;
31.    signal a0_4_0 : std_logic;
32.    signal a0_2_2 : std_logic;
33.    signal a0_3_2 : std_logic;
34.    signal a0_4_1 : std_logic;
35.    signal a0_5_0 : std_logic;
36.    signal a0_3_3 : std_logic;
37.    signal a0_4_2 : std_logic;
38.    signal a0_5_1 : std_logic;
39.    signal a0_6_0 : std_logic;
40.
41.
42.   -- Señales de salida
43.    signal RB0 : std_logic;
44.    signal RB1 : std_logic;
45.    signal RB2 : std_logic;
46.    signal RB3 : std_logic;
47.    signal RB4 : std_logic;
48.    signal RB5 : std_logic;
49.    signal RB6 : std_logic;
50.    signal RB7 : std_logic;
```

```
51.
52.   -- Etapa 1
53.    signal a1_0_0 : std_logic;
54.    signal a1_1_0 : std_logic;
55.    signal a1_2_0 : std_logic;
56.    signal a1_2_1 : std_logic;
57.    signal a1_3_0 : std_logic;
58.    signal a1_3_1 : std_logic;
59.    signal a1_3_2 : std_logic;
60.    signal a1_4_0 : std_logic;
61.    signal a1_4_1 : std_logic;
62.    signal a1_5_0 : std_logic;
63.    signal a1_5_1 : std_logic;
64.    signal a1_6_0 : std_logic;
65.    signal a1_6_1 : std_logic;
66.
67.
68.   -- Etapa 2
69.    signal a2_0_0 : std_logic;
70.    signal a2_1_0 : std_logic;
71.    signal a2_2_0 : std_logic;
72.    signal a2_3_0 : std_logic;
73.    signal a2_3_1 : std_logic;
74.    signal a2_4_0 : std_logic;
75.    signal a2_4_1 : std_logic;
76.    signal a2_4_2 : std_logic;
77.    signal a2_5_0 : std_logic;
78.    signal a2_5_1 : std_logic;
79.    signal a2_5_2 : std_logic;
80.    signal a2_6_0 : std_logic;
81.    signal a2_6_1 : std_logic;
82.    signal a2_6_2 : std_logic;
83.    signal a2_7_0 : std_logic;
84.    signal a2_7_1 : std_logic;
85.    signal a3_0_0 : std_logic;
86.    signal a3_0_1 : std_logic;
87.    signal a3_1_0 : std_logic;
88.    signal a3_1_1 : std_logic;
89.    signal a3_2_0 : std_logic;
90.    signal a3_2_1 : std_logic;
91.    signal a3_3_0 : std_logic;
92.    signal a3_3_1 : std_logic;
93.    signal a3_4_0 : std_logic;
94.    signal a3_4_1 : std_logic;
95.    signal a3_5_0 : std_logic;
96.    signal a3_5_1 : std_logic;
97.    signal a3_6_0 : std_logic;
98.    signal a3_6_1 : std_logic;
99.    signal a3_7_0 : std_logic;
100.   signal a3_7_1 : std_logic;
101.   signal a3_8_0 : std_logic;
102.   signal a3_8_1 : std_logic;
103.
```

```
104. begin
105.   a0_0_0 <= DatoA(0) and DatoB(0);
106.   a0_1_0 <= DatoA(1) and DatoB(0);
107.   a0_2_0 <= DatoA(2) and DatoB(0);
108.   a0_3_0 <= DatoA(3) and DatoB(0);
109.   a0_1_1 <= DatoA(0) and DatoB(1);
110.   a0_2_1 <= DatoA(1) and DatoB(1);
111.   a0_3_1 <= DatoA(2) and DatoB(1);
112.   a0_4_0 <= DatoA(3) and DatoB(1);
113.   a0_2_2 <= DatoA(0) and DatoB(2);
114.   a0_3_2 <= DatoA(1) and DatoB(2);
115.   a0_4_1 <= DatoA(2) and DatoB(2);
116.   a0_5_0 <= DatoA(3) and DatoB(2);
117.   a0_3_3 <= DatoA(0) and DatoB(3);
118.   a0_4_2 <= DatoA(1) and DatoB(3);
119.   a0_5_1 <= DatoA(2) and DatoB(3);
120.   a0_6_0 <= DatoA(3) and DatoB(3);
121.
122.
123.  -- sumatoiros
124.
125.
126.
127.  --Etapa 0
128. RB0 <= a0_0_0;
129. H0:H_adder port map(a1_1_0, a1_2_0,a0_1_0,a0_1_1);
130. RB1 <= a1_1_0;
131. F0:F_adder port map (a1_2_1,
    a1_3_0,a0_2_0,a0_2_1,a0_2_2);
132. F1:F_adder port map (a1_3_1,
    a1_4_0,a0_3_0,a0_3_1,a0_3_2);
133. a1_3_2 <= a0_3_3;
134. F2:F_adder port map (a1_4_1,
    a1_5_0,a0_4_0,a0_4_1,a0_4_2);
135. H1:H_adder port map(a1_5_1, a1_6_0,a0_5_0,a0_5_1);
136. a1_6_1 <= a0_6_0;
137.
138.
139.  -- Etapa 1
140. H2:H_adder port map(a2_2_0, a2_3_0, a1_2_0, a1_2_1);
141. RB2 <= a2_2_0;
142. F3:F_adder port map(a2_3_1,
    a2_4_0,a1_3_0,a1_3_1,a1_3_2);
143. H3:H_adder port map(a2_4_1, a2_5_0, a1_4_0, a1_4_1);
144. H4:H_adder port map(a2_5_1, a2_6_0, a1_5_0, a1_5_1);
145. H5:H_adder port map(a2_6_1, a2_7_0, a1_6_0, a1_6_1);
146.
147.  -- Etapa 2
148. H6:H_adder port map(RB3, a3_4_0, a2_3_0, a2_3_1);
149. F4:F_adder port map(RB4, a3_5_0,a2_4_0,a2_4_1,
    a3_4_0);
150. F5:F_adder port map(RB5, a3_6_0,a2_5_0,a2_5_1,
    a3_5_0);
```

```
151. F6:F_adder port map(RB6, a3_7_0,a2_6_0,a2_6_1,
   a3_6_0);
152. H7:H_adder port map(RB7, a3_8_0, a2_7_0, a3_7_0);
153.
154.
155.  -- concatenado
156.   resultado <= RB7 & RB6 & RB5 & RB4 & RB3 & RB2 & RB1
    & RB0;
157. end Behavioral;
```

Anexo 3: Descripción de los circuitos multiplicadores en VHDL.

Anexo 4: Código C++ de aplicación de generación del circuito multiplicación de Árbol de Wallace

```
1. #include <iostream>
2. #include <fstream>
3. using namespace std;
4.
5. void generar_sumas(int nbits);
6. void generar_senales(int nbitsR, int nbitsM);
7. void generar_senales_parciales (int nbits) ;
8. void multiplicaciones_parciales(int nbits);
9. ofstream myfile;
10.
11.   int main () {
12.     //Declaraciones
13.
14.     int nbits, nbitsM, nbitsR;
15.      //Petición de número de bits
16.     cout << "Numeros de bits de los datos a
   multiplicar: ";
17.     cin >> nbits;
18.     // Código bloc de notas
19.     int calculo, resultados, f_sum, h_sum;
20.     int carrys=0;
21.     int carrys_a=0;
22.
23.     //def bits
24.     nbitsM=nbits-1;
25.     nbitsR=nbits*2-1;
26.
27.     // Código VHDL
28.     myfile.open ("Codigo5_1.txt");
29.     myfile << "library IEEE;\n";
30.     myfile << "use IEEE.STD_LOGIC_1164.ALL;\n";
31.     myfile << "USE IEEE.STD_LOGIC_ARITH.ALL;\n";
32.     myfile << "USE IEEE.STD_LOGIC_UNSIGNED.ALL;\n\n";
33.
34.     myfile << "entity Multiplicador is\n";
35.     myfile << "Port (  DatoA, DatoB : in
   STD_LOGIC_VECTOR(" << nbitsM << " DOWNTO 0);\n";
36.     myfile << "   Resultado : out
   STD_LOGIC_VECTOR(" << nbitsR << " DOWNTO 0));\n";
37.     myfile << "end Multiplicador;\n\n";
38.     myfile << "architecture Behavioral of
   Multiplicador is\n";
39.
40.     myfile << "component F_adder is\n";
41.       myfile << "port(  res : out STD_LOGIC;\n";
42.       myfile << "   carry : out STD_LOGIC;\n";
43.     myfile << "   a : in STD_LOGIC;\n";
44.     myfile << "   b : in STD_LOGIC;\n";
45.     myfile << "   c : in STD_LOGIC);\n";
```

```
46.      myfile << "end component;\n";
47.      myfile << "component H_adder is\n";
48.        myfile << "port(  res : out STD_LOGIC;\n";
49.        myfile << "   carry : out STD_LOGIC;\n";
50.      myfile << "   a : in STD_LOGIC;\n";
51.      myfile << "   b : in STD_LOGIC);\n";
52.      myfile << "end component;\n";
53.
54.
55.      generar_senales_parciales (nbits);
56.      generar_senales(nbitsR, nbitsM);
57.
58.
59.      myfile << "\nbegin\n";
60.
61.      multiplicaciones_parciales(nbits);
62.      myfile << "\n\n -- sumatoiros \n\n";
63.      generar_sumas(nbits);
64.
65.
66.
67.      myfile << "end Behavioral;";
68.      myfile.close();
69.      cout << "Codigo Realizado";
70.        return 0;
71.    }
72.
73.
74.    void generar_senales(int nbitsZ, int nbitsM)
75.    {
76.      int nbits;
77.      nbits=nbitsM+1;
78.      int etapa=0;
79.      int nbitsR=nbitsZ+1;
80.      int calculo, resultados, f_sum, h_sum, mayor, a;
81.      int carrys=0;
82.        int carrys_a=0, matrix1[nbitsR][6],
   matrix2[nbitsR][6];
83.      int n=0;
84.       // Etapa 0
85.
86.      cout << "\n\n Etapa 0" ;
87.      myfile << "\n\n -- Etapa "<< etapa << endl ;
88.      for (int t=0; t<nbitsR; t++)
89.        {
90.        if(t<=nbitsM)
91.        {
92.          matrix1[t][0]=t;
93.          calculo=t+1;
94.          matrix1[t][1]=calculo;
95.          resultados=calculo+carrys_a;
96.          matrix1[t][2]=resultados;
97.          f_sum= (resultados)/3;
```

```
98.           h_sum= ((resultados)%3)/2;
99.         matrix1[t][3]=f_sum;
100.        matrix1[t][4]=h_sum;
101.        carrys=f_sum+h_sum;
102.        matrix1[t][5]=carrys;
103.      }
104.      else
105.      {
106.        matrix1[t][0]=t;
107.        calculo=nbitsR-t-1;
108.        matrix1[t][1]=calculo;
109.        resultados=calculo;
110.        matrix1[t][2]=resultados;
111.        if(resultados < 2)
112.        {
113.
114.          f_sum=0;
115.          h_sum=0;
116.          matrix1[t][3]=0;
117.          matrix1[t][4]=0;
118.          carrys=0;
119.        }
120.        else
121.        {
122.          f_sum= (resultados)/3;
123.            h_sum= ((resultados)%3)/2;
124.          matrix1[t][3]=f_sum;
125.          matrix1[t][4]=h_sum;
126.          carrys=f_sum+h_sum;
127.        }
128.        matrix1[t][5]=carrys;
129.      }
130.
131.    }
132.
133.    cout << "\n";
134.    cout<<"n  numC  nbt fadder  hadder  carrys\n";
135.    for (int i = 0; i < nbitsR; i++)
136.      {
137.          for (int j = 0; j < 6; j++)
138.          {
139.              cout<<matrix1[i][j]<<"  ";
140.          }
141.          cout << "\n";
142.      }
143.
144.      int p;
145.    for (int j = 1; j < nbitsR; j++)
146.    {
147.      p=(matrix1[j-
   1][5]+matrix1[j][4]+matrix1[j][3]+(matrix1[j][2]-
   matrix1[j][3]*3-matrix1[j][4]*2));
148.      if(p>a)
```

```
149.       {
150.         a=p;
151.       }
152.     }
153.
154.
155.     etapa++;
156.     mayor=a;
157.     // etapa 1 y siguientes
158.
159.       while( mayor != 2 && mayor !=3 && mayor !=4)
160.       {
161.         myfile << "\n\n -- Etapa "<< etapa << endl ;
162.         cout << "\n\n Etapa "<< etapa ;
163.         a=0;
164.         int Q;
165.           for (int t=0; t<nbitsR; t++)
166.           {
167.           carrys_a=matrix1[t-1][5];
168.           Q=(matrix1[t][2]-matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4]+matrix1[t
   -1][5];
169.           if(Q>2)
170.           {
171.             matrix2[t][0]=t;
172.             if(t==0)
173.             {
174.               matrix2[t][1]=matrix1[t][1];
175.               matrix2[t][2]=matrix1[t][2];
176.             }
177.             else
178.             {
179.               matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4];
180.               matrix2[t][2]=carrys_a+matrix2[t][1];
181.             }
182.
183.             f_sum= (matrix2[t][2])/3;
184.             h_sum= ((matrix2[t][2])%3)/2;
185.             matrix2[t][3]=f_sum;
186.             matrix2[t][4]=h_sum;
187.             carrys=f_sum+h_sum;
188.             matrix2[t][5]=carrys;
189.           }
190.           else if(Q==2 && t<=(nbitsR/2))
191.           {
192.             matrix2[t][0]=t;
193.             if(t==0)
194.             {
195.               matrix2[t][1]=matrix1[t][1];
196.               matrix2[t][2]=matrix1[t][2];
197.             }
```

```
198.            else
199.            {
200.              matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4];
201.              matrix2[t][2]=carrys_a+matrix2[t][1];
202.            }
203.
204.            f_sum= (matrix2[t][2])/3;
205.            h_sum= ((matrix2[t][2])%3)/2;
206.            matrix2[t][3]=f_sum;
207.            matrix2[t][4]=h_sum;
208.            carrys=f_sum+h_sum;
209.            matrix2[t][5]=carrys;
210.          }
211.          else if(Q==2 && t>(nbitsR/2))
212.          {
213.            f_sum=0;
214.            h_sum=0;
215.            matrix2[t][0]=matrix1[t][0];
216.            matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4];
217.            matrix2[t][2]=Q;
218.            matrix2[t][3]=0;
219.            matrix2[t][4]=0;
220.            carrys=0;
221.            matrix2[t][5]=0;
222.          }
223.          else
224.          {
225.            f_sum=0;
226.            h_sum=0;
227.            matrix2[t][0]=matrix1[t][0];
228.            matrix2[t][1]=Q;
229.            matrix2[t][2]=matrix2[t][1];
230.            matrix2[t][3]=0;
231.            carrys=0;
232.            matrix2[t][4]=0;
233.            matrix2[t][5]=0;
234.          }
235.
236.          for (int r = 0; r < matrix2[t][2]; r++)
237.              {
238.                  myfile << " signal
   a"<< etapa << "_" << t << "_" << r <<" :
   std_logic;\n"; // primer 0 identifica la etapa 0 , t
   columna, e orden de dato
239.              }
240.
241.
242.        }
243.        cout << "\n";
```

Anexo 4: Código C++ de aplicación de generación del circuito multiplicación de Árbol de Wallace

```
244.         cout<<"n  numC  nbt
   fadder  hadder  carrys\n";
245.         for (int i = 0; i < nbitsR; i++)
246.           {
247.               for (int j = 0; j < 6; j++)
248.               {
249.                   cout<<matrix2[i][j]<<"  ";
250.               }
251.               cout << "\n";
252.           }
253.         int p;
254.         for (int j = 1; j < nbitsR; j++)
255.           {
256.           p=(matrix2[j-
   1][5]+matrix2[j][4]+matrix2[j][3]+(matrix2[j][2]-
   matrix2[j][3]*3-matrix2[j][4]*2));
257.           if(p>a)
258.           {
259.             a=p;
260.           }
261.           }
262.           for (int i = 0; i < nbitsR; i++)
263.           {
264.               for (int j = 0; j < 6; j++)
265.               {
266.               matrix1[i][j]=matrix2[i][j];
267.               }
268.           }
269.           mayor=a;
270.         etapa= etapa +1;
271.       }
272.
273.       // etapa para cuando el maximo es 4 ejecucion
   especial
274.       if( mayor == 4)
275.       {
276.         myfile << "\n\n -- Etapa "<< etapa << endl ;
277.         for (int t=0; t<nbitsR; t++)
278.         {
279.           carrys_a=matrix1[t-1][5];
280.           matrix2[t][0]=t;
281.           if(t==0)
282.           {
283.             matrix2[t][1]=matrix1[t][1];
284.             matrix2[t][2]=matrix1[t][2];
285.           }
286.           else
287.           {
288.             matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4];
289.             matrix2[t][2]=carrys_a+matrix2[t][1];
290.           }
```

```
291.           if(t>(nbitsR/2))
292.           {
293.             if(matrix2[t][2]==3)
294.             {
295.               f_sum= 0;
296.               h_sum= 1;
297.             }
298.             else if(matrix2[t][2]==2)
299.             {
300.               f_sum= 0;
301.               h_sum= 0;
302.             }
303.             else if(matrix2[t][2]>3)
304.             {
305.               f_sum= (matrix2[t][2])/3;
306.               h_sum= ((matrix2[t][2])%3)/2;
307.             }
308.             else
309.             {
310.               f_sum= 0;
311.               h_sum= 0;
312.             }
313.           }
314.           else
315.           {
316.             f_sum= (matrix2[t][2])/3;
317.             h_sum= ((matrix2[t][2])%3)/2;
318.           }
319.           matrix2[t][3]=f_sum;
320.           matrix2[t][4]=h_sum;
321.           carrys=f_sum+h_sum;
322.           matrix2[t][5]=carrys;
323.           for (int r = 0; r < matrix2[t][2]; r++)
324.               {
325.                   myfile << " signal
   a"<< etapa << "_" << t << "_" << r <<" :
   std_logic;\n"; // primer 0 identifica la etapa 0 , t
   columna, e orden de dato
326.               }
327.         }
328.         cout << "\n Etapa " << etapa << endl;
329.         cout<<"n  numC  nbt
   fadder  hadder  carrys\n";
330.         for (int i = 0; i < nbitsR; i++)
331.         {
332.           for (int j = 0; j < 6; j++)
333.           {
334.               cout<<matrix2[i][j]<<"  ";
335.           }
336.           cout << "\n";
337.         }
338.           for (int i = 0; i < nbitsR; i++)
339.           {
```

Anexo 4: Código C++ de aplicación de generación del circuito multiplicación de Árbol de Wallace

```
340.                for (int j = 0; j < 6; j++)
341.                {
342.                matrix1[i][j]=matrix2[i][j];
343.                }
344.            }
345.          a=0;
346.          int p;
347.          for (int j = 1; j < nbitsR; j++)
348.            {
349.            p=(matrix2[j-
   1][5]+matrix2[j][4]+matrix2[j][3]+(matrix2[j][2]-
   matrix2[j][3]*3-matrix2[j][4]*2));
350.            if(p>a)
351.            {
352.              a=p;
353.            }
354.            }
355.            mayor=a;
356.          etapa= etapa +1;
357.        }
358.
359.        // etapa para cuando el maximo es 3 ejecucion
   especial
360.        if( mayor == 3)
361.        {
362.          myfile << "\n\n -- Etapa "<< etapa << endl ;
363.          for (int t=0; t<nbitsR; t++)
364.          {
365.            carrys_a=matrix1[t-1][5];
366.            matrix2[t][0]=t;
367.            if(t==0)
368.            {
369.              matrix2[t][1]=matrix1[t][1];
370.              matrix2[t][2]=matrix1[t][2];
371.            }
372.            else
373.            {
374.              matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4];
375.              matrix2[t][2]=carrys_a+matrix2[t][1];
376.            }
377.
378.            f_sum= (matrix2[t][2])/3;
379.            h_sum= ((matrix2[t][2])%3)/2;
380.            matrix2[t][3]=f_sum;
381.            matrix2[t][4]=h_sum;
382.            carrys=f_sum+h_sum;
383.            matrix2[t][5]=carrys;
384.
385.            for (int r = 0; r < matrix2[t][2]; r++)
386.                {
```

Anexo 4: Código C++ de aplicación de generación del circuito multiplicación de Árbol de Wallace

```
387.                    myfile << " signal
   a"<< etapa << "_" << t << "_" << r <<" :
   std_logic;\n"; // primer 0 identifica la etapa 0 , t
   columna, e orden de dato
388.                }
389.          }
390.
391.          cout << "\n Etapa " << etapa << endl;
392.          cout<<"n  numC  nbt
   fadder  hadder  carrys\n";
393.          for (int i = 0; i < nbitsR; i++)
394.          {
395.            for (int j = 0; j < 6; j++)
396.            {
397.                cout<<matrix2[i][j]<<"  ";
398.            }
399.            cout << "\n";
400.          }
401.            for (int i = 0; i < nbitsR; i++)
402.            {
403.                for (int j = 0; j < 6; j++)
404.                {
405.                matrix1[i][j]=matrix2[i][j];
406.                }
407.            }
408.
409.          a=0;
410.          int p;
411.          for (int j = 1; j < nbitsR; j++)
412.            {
413.            p=(matrix2[j-
   1][5]+matrix2[j][4]+matrix2[j][3]+(matrix2[j][2]-
   matrix2[j][3]*3-matrix2[j][4]*2));
414.            if(p>a)
415.            {
416.              a=p;
417.            }
418.            }
419.            mayor=a;
420.          etapa= etapa +1;
421.        }
422.        if( mayor == 2)
423.        {
424.          // etapa final
425.          myfile << "\n\n -- Etapa "<< etapa << endl ;
426.          for (int t=0; t<nbitsR; t++)
427.          {
428.            carrys_a=matrix1[t-1][5];
429.            matrix2[t][0]=t;
430.            if(t==0)
431.            {
432.              matrix2[t][1]=matrix1[t][1];
433.              matrix2[t][2]=matrix1[t][2];
```

```
434.           }
435.           else
436.           {
437.             matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4]+carrys_a;
438.             matrix2[t][2]=matrix2[t][1]+carrys;
439.           }
440.
441.           f_sum= (matrix2[t][2])/3;
442.           h_sum= ((matrix2[t][2])%3)/2;
443.           matrix2[t][3]=f_sum;
444.           matrix2[t][4]=h_sum;
445.           carrys=f_sum+h_sum;
446.           matrix2[t][5]=carrys;
447.
448.
449.           for (int r = 0; r < matrix2[t][2]; r++)
450.               {
451.                   myfile << " signal
   a"<< etapa << "_" << t << "_" << r <<" :
   std_logic;\n"; // primer 0 identifica la etapa 0 , t
   columna, e orden de dato
452.               }
453.         }
454.         cout << "\n Etapa" << etapa << endl;
455.         cout<<"n  numC  nbt
   fadder  hadder  carrys\n";
456.         for (int i = 0; i < nbitsR; i++)
457.         {
458.           for (int j = 0; j < 6; j++)
459.           {
460.               cout<<matrix2[i][j]<<"  ";
461.           }
462.           cout << "\n";
463.         }
464.           for (int i = 0; i < nbitsR; i++)
465.           {
466.               for (int j = 0; j < 6; j++)
467.               {
468.               matrix1[i][j]=matrix2[i][j];
469.               }
470.         }
471.         for (int x =0; x<=etapa; x++)
472.         {
473.           myfile << "signal RB" << x <<" :
   std_logic;\n";
474.         }
475.         etapa= etapa +1;
476.         myfile << "signal primer_operando :
   std_logic_vector (" << nbitsZ-etapa << " downto
   0);\n";
```

```
477.         myfile << "signal segundo_operando :
   std_logic_vector (" << nbitsZ-1-etapa << " downto
   0);\n";
478.         myfile << "signal resultado_operando :
   std_logic_vector (" << nbitsZ-etapa << " downto
   0);\n";
479.
480.       }
481.
482.   }
483.
484.
485.   void generar_sumas(int nbits)
486.   {
487.     int nbitsM=nbits-1;
488.     int nbitsZ=nbits*2-1;
489.     //nbits=nbitsM+1;
490.     int n=0;
491.     int etapa=0;
492.     int nbitsR=nbitsZ+1;
493.     int calculo, resultados, f_sum, h_sum, mayor, a;
494.     int carrys=0;
495.       int carrys_a=0, matrix1[nbitsR][6],
   matrix2[nbitsR][6];
496.     int sumatorioFA=0, sumatorioHA=0;
497.      // Etapa 0
498.
499.     myfile << "\n\n --Etapa 0\n" ;
500.     myfile <<"RB0 <= a0_0_0;\n";
501.     for (int t=0; t<nbitsR; t++)
502.       {
503.       if(t<=nbitsM)
504.       {
505.         matrix1[t][0]=t;
506.         calculo=t+1;
507.         matrix1[t][1]=calculo;
508.         resultados=calculo+carrys_a;
509.         matrix1[t][2]=resultados;
510.         f_sum= (resultados)/3;
511.           h_sum= ((resultados)%3)/2;
512.         matrix1[t][3]=f_sum;
513.         matrix1[t][4]=h_sum;
514.         carrys=f_sum+h_sum;
515.         matrix1[t][5]=carrys;
516.       }
517.       else
518.       {
519.         matrix1[t][0]=t;
520.         calculo=nbitsR-t-1;
521.         matrix1[t][1]=calculo;
522.         resultados=calculo;
523.         matrix1[t][2]=resultados;
524.         if(resultados < 2)
```

```
525.         {
526.
527.           f_sum=0;
528.           h_sum=0;
529.           matrix1[t][3]=0;
530.           matrix1[t][4]=0;
531.           carrys=0;
532.         }
533.         else
534.         {
535.           f_sum= (resultados)/3;
536.             h_sum= ((resultados)%3)/2;
537.           matrix1[t][3]=f_sum;
538.           matrix1[t][4]=h_sum;
539.           carrys=f_sum+h_sum;
540.         }
541.         matrix1[t][5]=carrys;
542.       }
543.
544.       int h=0;
545.       int k=0;
546.       int k1=0;
547.       if(f_sum!=0)
548.       {
549.         for(h; h < f_sum; h++)
550.         {
551.           myfile << "F" <<sumatorioFA <<":F_adder
   port map
   (a" << etapa+1 << "_" << t << "_" << h+matrix1[t-
   1][5] <<", a" << etapa+1 << "_" << t+1 << "_" << h;
552.           for(k; k< 3+k1; k++)
553.           {
554.             myfile << ",a" << etapa << "_" << t <<"_"
    << k;
555.           }
556.           k1=k;
557.           myfile <<");\n";
558.           sumatorioFA+=1;
559.         }
560.       }
561.       if(h_sum!=0)
562.       {
563.         for(h; h < f_sum+h_sum; h++)
564.         {
565.           myfile << "H" <<sumatorioHA <<":H_adder
   port
   map(a" << etapa+1 << "_" << t << "_" << h+matrix1[t-
   1][5] <<", a" << etapa+1 << "_" << t+1 << "_" << h;
566.           for(k; k< 2+k1; k++)
567.               {
568.                 myfile << ",a" << etapa << "_" << t <
   <"_" << k;
569.               }
```

```
570.           k1=k;
571.           myfile <<");\n";
572.           sumatorioHA+=1;
573.         }
574.       }
575.       int w,o;
576.       w=matrix1[t][2]-3*f_sum-2*h_sum;
577.       o=3*f_sum+2*h_sum;
578.       if(((h_sum!=1 && f_sum!=0) || (h_sum!=0 && f_su
   m!=1)) && matrix1[t-1][5]==0 && t<(nbitsR/2))
579.       {
580.         myfile <<"RB" << t << " <=
   a" << etapa+1 << "_" << t << "_" << h-1 <<";\n";
581.         n+=1;
582.       }
583.       if(w!=0 && t>n)
584.       {
585.         for(h; h < f_sum+h_sum+w; h++)
586.         {
587.           myfile <<"a" << etapa+1 << "_" << t << "_"
   << h+matrix1[t-1][5] <<" <=
   a" << etapa << "_" << t << "_" << o <<";\n";
588.           o+=1;
589.         }
590.       }
591.
592.     }
593.
594.       int p;
595.     for (int j = 1; j < nbitsR; j++)
596.     {
597.       p=(matrix1[j-
   1][5]+matrix1[j][4]+matrix1[j][3]+(matrix1[j][2]-
   matrix1[j][3]*3-matrix1[j][4]*2));
598.       if(p>a)
599.       {
600.         a=p;
601.       }
602.     }
603.
604.
605.     etapa++;
606.     mayor=a;
607.     // etapa 1 y siguientes
608.
609.       while( mayor != 2 && mayor !=3 && mayor !=4)
610.       {
611.         myfile << "\n -- Etapa "<< etapa << endl ;
612.         a=0;
613.         int Q;
614.           for (int t=0; t<nbitsR; t++)
615.           {
616.           carrys_a=matrix1[t-1][5];
```

```
617.          if(t==0)
618.          {
619.              Q=0;
620.          }
621.          else
622.          {
623.            Q=(matrix1[t][2]-matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4]+matrix1[t
   -1][5];
624.          }
625.
626.          if(Q>2)
627.          {
628.            matrix2[t][0]=t;
629.            if(t==0)
630.            {
631.              matrix2[t][1]=matrix1[t][1];
632.              matrix2[t][2]=matrix1[t][2];
633.            }
634.            else
635.            {
636.              matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4];
637.              matrix2[t][2]=carrys_a+matrix2[t][1];
638.            }
639.
640.            f_sum= (matrix2[t][2])/3;
641.            h_sum= ((matrix2[t][2])%3)/2;
642.            matrix2[t][3]=f_sum;
643.            matrix2[t][4]=h_sum;
644.            carrys=f_sum+h_sum;
645.            matrix2[t][5]=carrys;
646.          }
647.          else if(Q==2 && t<=(nbitsR/2))
648.          {
649.            matrix2[t][0]=t;
650.            if(t==0)
651.            {
652.              matrix2[t][1]=matrix1[t][1];
653.              matrix2[t][2]=matrix1[t][2];
654.            }
655.            else
656.            {
657.              matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4];
658.              matrix2[t][2]=carrys_a+matrix2[t][1];
659.            }
660.
661.            f_sum= (matrix2[t][2])/3;
662.            h_sum= ((matrix2[t][2])%3)/2;
663.            matrix2[t][3]=f_sum;
```

```
664.            matrix2[t][4]=h_sum;
665.            carrys=f_sum+h_sum;
666.            matrix2[t][5]=carrys;
667.          }
668.          else if(Q==2 && t>(nbitsR/2))
669.          {
670.            f_sum=0;
671.            h_sum=0;
672.            matrix2[t][0]=matrix1[t][0];
673.            matrix2[t][1]=Q;
674.            matrix2[t][2]=matrix2[t][1];
675.            matrix2[t][3]=0;
676.            matrix2[t][4]=0;
677.            carrys=0;
678.            matrix2[t][5]=0;
679.          }
680.          else
681.          {
682.            f_sum=0;
683.            h_sum=0;
684.            matrix2[t][0]=matrix1[t][0];
685.            matrix2[t][1]=Q;
686.            matrix2[t][2]=matrix2[t][1];
687.            matrix2[t][3]=0;
688.            carrys=0;
689.            matrix2[t][4]=0;
690.            matrix2[t][5]=0;
691.          }
692.
693.          int h=0;
694.          int k=0;
695.          int k1=0;
696.          if(f_sum!=0)
697.          {
698.            for(h; h < f_sum; h++)
699.            {
700.              myfile << "F" <<sumatorioFA <<":F_adder
   port
   map(a" << etapa+1 << "_" << t << "_" << h+matrix2[t-
   1][5] <<", a" << etapa+1 << "_" << t+1 << "_" << h;
701.              for(k; k< 3+k1; k++)
702.              {
703.                myfile << ",a" << etapa << "_" << t <
   <"_" << k;
704.              }
705.              k1=k;
706.              myfile <<");\n";
707.              sumatorioFA+=1;
708.            }
709.          }
710.          if(h_sum!=0)
711.          {
712.            for(h; h < f_sum+h_sum; h++)
```

```
713.            {
714.              myfile << "H" <<sumatorioHA <<":H_adder
   port
   map(a" << etapa+1 << "_" << t << "_" << h+matrix2[t-
   1][5] <<", a" << etapa+1 << "_" << t+1 << "_" << h;
715.              for(k; k< 2+k1; k++)
716.              {
717.                myfile << ",a" << etapa << "_" << t <
   <"_" << k;
718.              }
719.              k1=k;
720.              myfile <<");\n";
721.              sumatorioHA+=1;
722.            }
723.          }
724.          int o,w;
725.          o=3*f_sum+2*h_sum;
726.          w=Q-o;
727.          if(((h_sum!=1 && f_sum!=0) || (h_sum!=0 &&
   f_sum!=1)) && matrix2[t-1][5]==0 && t<(nbitsR/2))
728.          {
729.            myfile <<"RB" << t << " <=
   a" << etapa+1 << "_" << t << "_" << h-1 <<";\n";
730.            n+=1;
731.          }
732.          if(w!=0 && t>n)
733.          {
734.            for(h; h < f_sum+h_sum+w; h++)
735.            {
736.              myfile <<"a" << etapa+1 << "_" << t <<
   "_" << h+matrix2[t-1][5] <<" <=
   a" << etapa << "_" << t << "_" << o <<";\n";
737.              o+=1;
738.            }
739.          }
740.        }
741.
742.        int p;
743.        for (int j = 1; j < nbitsR; j++)
744.          {
745.          p=(matrix2[j-
   1][5]+matrix2[j][4]+matrix2[j][3]+(matrix2[j][2]-
   matrix2[j][3]*3-matrix2[j][4]*2));
746.          if(p>a)
747.          {
748.            a=p;
749.          }
750.          }
751.          for (int i = 0; i < nbitsR; i++)
752.          {
753.              for (int j = 0; j < 6; j++)
754.              {
755.              matrix1[i][j]=matrix2[i][j];
```

```
756.                }
757.            }
758.            mayor=a;
759.          etapa= etapa +1;
760.        }
761.
762.        // etapa para cuando el maximo es 4 ejecucion
    especial
763.        if( mayor == 4)
764.        {
765.          myfile << "\n -- Etapa "<< etapa << endl ;
766.          for (int t=0; t<nbitsR; t++)
767.          {
768.            carrys_a=matrix1[t-1][5];
769.            matrix2[t][0]=t;
770.            if(t==0)
771.            {
772.              matrix2[t][1]=matrix1[t][1];
773.              matrix2[t][2]=matrix1[t][2];
774.            }
775.            else
776.            {
777.              matrix2[t][1]=(matrix1[t][2]-
    matrix1[t][3]*3-
    matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4];
778.              matrix2[t][2]=carrys_a+matrix2[t][1];
779.            }
780.            if(t>(nbitsR/2))
781.            {
782.              if(matrix2[t][2]==3)
783.              {
784.                f_sum= 0;
785.                h_sum= 1;
786.              }
787.              else if(matrix2[t][2]==2)
788.              {
789.                f_sum= 0;
790.                h_sum= 0;
791.              }
792.              else if(matrix2[t][2]>3)
793.              {
794.                f_sum= (matrix2[t][2])/3;
795.                h_sum= ((matrix2[t][2])%3)/2;
796.              }
797.              else
798.              {
799.                f_sum= 0;
800.                h_sum= 0;
801.              }
802.            }
803.            else
804.            {
805.              f_sum= (matrix2[t][2])/3;
```

```
806.            h_sum= ((matrix2[t][2])%3)/2;
807.          }
808.          matrix2[t][3]=f_sum;
809.          matrix2[t][4]=h_sum;
810.          carrys=f_sum+h_sum;
811.          matrix2[t][5]=carrys;
812.
813.
814.          int h=0;
815.          int k=0;
816.          int k1=0;
817.          if(f_sum!=0)
818.          {
819.            for(h; h < f_sum; h++)
820.            {
821.              myfile << "F" <<sumatorioFA <<":F_adder
   port
   map(a" << etapa+1 << "_" << t << "_" << h+matrix2[t-
   1][5] <<", a" << etapa+1 << "_" << t+1 << "_" << h;
822.              for(k; k< 3+k1; k++)
823.              {
824.                myfile << ",a" << etapa << "_" << t <
   <"_" << k;
825.              }
826.              k1=k;
827.              myfile <<");\n";
828.              sumatorioFA+=1;
829.            }
830.          }
831.          if(h_sum!=0)
832.          {
833.            for(h; h < f_sum+h_sum; h++)
834.            {
835.              myfile << "H" <<sumatorioHA <<":H_adder
   port
   map(a" << etapa+1 << "_" << t << "_" << h+matrix2[t-
   1][5] <<", a" << etapa+1 << "_" << t+1 << "_" << h;
836.              for(k; k< 2+k1; k++)
837.              {
838.                myfile << ",a" << etapa << "_" << t <
   <"_" << k;
839.              }
840.              k1=k;
841.              myfile <<");\n";
842.              sumatorioHA+=1;
843.            }
844.          }
845.          int o,w;
846.          int Q=(matrix1[t][2]-matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4]+matrix1[t
   -1][5];
847.          o=3*f_sum+2*h_sum;
848.          w=Q-o;
```

```
849.          if(((h_sum==1 && f_sum==0) || (h_sum==0 &&
   f_sum==1)) && matrix2[t-1][5]==0 && t<(nbitsR/2))
850.          {
851.            myfile <<"RB" << t << " <=
   a" << etapa+1 << "_" << t << "_" << h-1 <<";\n";
852.            n+=1;
853.          }
854.          if(w!=0 && t>n)
855.          {
856.            for(h; h < f_sum+h_sum+w; h++)
857.            {
858.              myfile <<"a" << etapa+1 << "_" << t <<
   "_" << h+matrix2[t-1][5] <<" <=
   a" << etapa << "_" << t << "_" << o <<";\n";
859.              o+=1;
860.            }
861.          }
862.        }
863.
864.          for (int i = 0; i < nbitsR; i++)
865.          {
866.              for (int j = 0; j < 6; j++)
867.              {
868.              matrix1[i][j]=matrix2[i][j];
869.              }
870.          }
871.        a=0;
872.        int p;
873.        for (int j = 1; j < nbitsR; j++)
874.          {
875.          p=(matrix2[j-
   1][5]+matrix2[j][4]+matrix2[j][3]+(matrix2[j][2]-
   matrix2[j][3]*3-matrix2[j][4]*2));
876.          if(p>a)
877.          {
878.            a=p;
879.          }
880.          }
881.          mayor=a;
882.        etapa= etapa +1;
883.      }
884.
885.      // etapa para cuando el maximo es 3 ejecucion
   especial
886.      if( mayor == 3)
887.      {
888.        myfile << "\n\n -- Etapa "<< etapa << endl ;
889.        for (int t=0; t<nbitsR; t++)
890.        {
891.          carrys_a=matrix1[t-1][5];
892.          matrix2[t][0]=t;
893.          if(t==0)
894.          {
```

```
895.            matrix2[t][1]=matrix1[t][1];
896.            matrix2[t][2]=matrix1[t][2];
897.          }
898.          else
899.          {
900.            matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4];
901.            matrix2[t][2]=carrys_a+matrix2[t][1];
902.          }
903.
904.          f_sum= (matrix2[t][2])/3;
905.          h_sum= ((matrix2[t][2])%3)/2;
906.          matrix2[t][3]=f_sum;
907.          matrix2[t][4]=h_sum;
908.          carrys=f_sum+h_sum;
909.          matrix2[t][5]=carrys;
910.
911.          int h=0;
912.          int k=0;
913.          int k1=0;
914.          if(f_sum!=0)
915.          {
916.            for(h; h < f_sum; h++)
917.            {
918.              myfile << "F" <<sumatorioFA <<":F_adder
   port
   map(a" << etapa+1 << "_" << t << "_" << h+matrix2[t-
   1][5] <<", a" << etapa+1 << "_" << t+1 << "_" << h;
919.              for(k; k< 3+k1; k++)
920.              {
921.                myfile << ",a" << etapa << "_" << t <
   <"_" << k;
922.              }
923.              k1=k;
924.              myfile <<");\n";
925.              sumatorioFA+=1;
926.            }
927.          }
928.          if(h_sum!=0)
929.          {
930.            for(h; h < f_sum+h_sum; h++)
931.            {
932.              myfile << "H" <<sumatorioHA <<":H_adder
   port
   map(a" << etapa+1 << "_" << t << "_" << h+matrix2[t-
   1][5] <<", a" << etapa+1 << "_" << t+1 << "_" << h;
933.              myfile << ",
   a" << etapa << "_" << t <<"_" << k << ",
   a" << etapa << "_" << t <<"_" << k+1;
934.              k1=k;
935.              myfile <<");\n";
936.              sumatorioHA+=1;
```

```
937.             }
938.           }
939.           int o,w;
940.           int Q=(matrix1[t][2]-matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4]+matrix1[t
   -1][5];
941.           o=3*f_sum+2*h_sum;
942.           w=Q-o;
943.           if(((h_sum!=1 && f_sum!=0) || (h_sum!=0 &&
   f_sum!=1)) && matrix2[t-1][5]==0 && t<(nbitsR/2))
944.           {
945.             myfile <<"RB" << t << " <=
   a" << etapa+1 << "_" << t << "_" << h-1 <<";\n";
946.             n+=1;
947.           }
948.           if(w!=0 && t>n)
949.           {
950.             for(h; h < f_sum+h_sum+w; h++)
951.             {
952.               myfile <<"a" << etapa+1 << "_" << t <<
   "_" << h+matrix2[t-1][5] <<" <=
   a" << etapa << "_" << t << "_" << o <<";\n";
953.               o+=1;
954.             }
955.           }
956.         }
957.
958.           for (int i = 0; i < nbitsR; i++)
959.           {
960.               for (int j = 0; j < 6; j++)
961.               {
962.               matrix1[i][j]=matrix2[i][j];
963.               }
964.           }
965.
966.         a=0;
967.         int p;
968.         for (int j = 1; j < nbitsR; j++)
969.           {
970.           p=(matrix2[j-
   1][5]+matrix2[j][4]+matrix2[j][3]+(matrix2[j][2]-
   matrix2[j][3]*3-matrix2[j][4]*2));
971.           if(p>a)
972.           {
973.             a=p;
974.           }
975.           }
976.           mayor=a;
977.         etapa= etapa +1;
978.       }
979.       if( mayor == 2)
980.       {
981.         // etapa final
```

```
982.        myfile << "\n -- Etapa "<< etapa << endl ;
983.        for (int t=0; t<nbitsR; t++)
984.        {
985.          carrys_a=matrix1[t-1][5];
986.          matrix2[t][0]=t;
987.          if(t==0)
988.          {
989.            matrix2[t][1]=matrix1[t][1];
990.            matrix2[t][2]=matrix1[t][2];
991.          }
992.          else
993.          {
994.            matrix2[t][1]=(matrix1[t][2]-
   matrix1[t][3]*3-
   matrix1[t][4]*2)+matrix1[t][3]+matrix1[t][4]+carrys_a;
995.            matrix2[t][2]=matrix2[t][1]+carrys;
996.          }
997.
998.          f_sum= (matrix2[t][2])/3;
999.          h_sum= ((matrix2[t][2])%3)/2;
1000.         matrix2[t][3]=f_sum;
1001.         matrix2[t][4]=h_sum;
1002.         carrys=f_sum+h_sum;
1003.         matrix2[t][5]=carrys;
1004.       }
1005.
1006.         for (int i = 0; i < nbitsR; i++)
1007.         {
1008.             for (int j = 0; j < 6; j++)
1009.             {
1010.             matrix1[i][j]=matrix2[i][j];
1011.             }
1012.         }
1013.       etapa= etapa +1;
1014.
1015.       myfile << "primer_operando <= a" << etapa-
   1 << "_" << nbitsZ << "_0";
1016.       for(int u=nbitsZ-1; u>=etapa; u--)
1017.       {
1018.         myfile <<" & a" << etapa-
   1 << "_" << u << "_0";
1019.       }
1020.
1021.       myfile << ";\n";
1022.
1023.       if (matrix2[nbitsZ-1][1] == 1)
1024.       {
1025.         myfile << "segundo_operando <= '0'";
1026.       }
1027.       else
1028.       {
1029.         myfile << "segundo_operando <= a" << etapa-
   1 << "_" << nbitsZ-1 << "_1";
```

```
1030.       }
1031.       for(int v=nbitsZ-2; v>=etapa; v--)
1032.       {
1033.         if (matrix2[v][1] == 1)
1034.         {
1035.           myfile <<" & '0'";
1036.         }
1037.         else
1038.         {
1039.           myfile <<" & a"<< etapa-
   1 << "_" << v << "_1";
1040.         }
1041.       }
1042.       myfile << ";\n";
1043.
1044.       myfile << "resultado_operando <=
   primer_operando + segundo_operando;\n";
1045.       myfile << "resultado <= resultado_operando";
1046.       for(int g= etapa-1; g>=0;g--)
1047.       {
1048.         myfile << " & RB"<<g;
1049.       }
1050.
1051.       myfile << ";\n";
1052.     }
1053.
1054. }
1055.
1056. void generar_senales_parciales (int nbits) {
1057.
1058.   int nbitsM=nbits-1;
1059.   myfile <<"\n";
1060.   for (int i=0; i<=nbitsM; i++)
1061.     {
1062.     for(int j=0; j <= nbitsM; j++)
1063.       {
1064.           if( (i+j) <= nbitsM )
1065.           {
1066.             myfile << " signal
   a0_"<< (i+j) << "_" << i <<" : std_logic;\n"; //
   primer 0 identifica la etapa 0 , t columna, e orden de
   dato
1067.           }
1068.           else
1069.           {
1070.             int valor;
1071.             valor=(i+nbitsM)-(i+j);
1072.             myfile << " signal
   a0_"<< (i+j) << "_" << valor <<" : std_logic;\n"; //
   primer 0 identifica la etapa 0 , t columna, e orden de
   dato
1073.           }
1074.       }
```

```
1075.    }
1076. }
1077.
1078. void multiplicaciones_parciales (int nbits) {
1079.
1080.    int nbitsM=nbits-1;
1081.
1082.    for (int i=0; i<=nbitsM; i++)
1083.      {
1084.      for(int j=0; j <= nbitsM; j++)
1085.        {
1086.            if( (i+j) <= nbitsM )
1087.            {
1088.              myfile << "
   a0_"<< (i+j) << "_" << i << " <= DatoA("<< j <<") and
   DatoB(" << i <<");\n";
1089.            }
1090.            else
1091.            {
1092.              int valor;
1093.              valor=(i+nbitsM)-(i+j);
1094.              myfile << "
   a0_"<< (i+j) << "_" << valor << " <= DatoA("<< j <<")
   and DatoB(" << i <<");\n";
1095.            }
1096.        }
1097.    }
1098. }
```

Bibliografía

[1] David Patterson y Andrew Waterman, *Guía práctica de RISC-V*, Primera. [En línea]. Disponible en: http://riscvbook.com/spanish/guia-practica-de-risc-v-1.0.5.pdf