

Reporte de practica 6

González Pardo Adrian

Marzo 2020

1. Código VHDL

1.1. Mux de 16 canales

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity muxGral is
5     Port (chanel_0:in std_logic_vector(15 downto 0);
6           chanel_1:in std_logic_vector(15 downto 0);
7           chanel_2:in std_logic_vector(15 downto 0);
8           chanel_3:in std_logic_vector(15 downto 0);
9           chanel_4:in std_logic_vector(15 downto 0);
10          chanel_5:in std_logic_vector(15 downto 0);
11          chanel_6:in std_logic_vector(15 downto 0);
12          chanel_7:in std_logic_vector(15 downto 0);
13          chanel_8:in std_logic_vector(15 downto 0);
14          chanel_9:in std_logic_vector(15 downto 0);
15          chanel_10:in std_logic_vector(15 downto 0);
16          chanel_11:in std_logic_vector(15 downto 0);
17          chanel_12:in std_logic_vector(15 downto 0);
18          chanel_13:in std_logic_vector(15 downto 0);
19          chanel_14:in std_logic_vector(15 downto 0);
20          chanel_15:in std_logic_vector(15 downto 0);
21          selectMux: in STD_LOGIC_VECTOR (3 downto 0);
22          outMux :out std_logic_vector(15 downto 0) );
23 end muxGral;
24
25 architecture Behavioral of muxGral is
26 begin
27     process(selectMux)
28     begin
29         case selectMux is
30             when "0000" =>outMux<= chanel_0;
31             when "0001" =>outMux<= chanel_1;
32             when "0010" =>outMux<= chanel_2;
33             when "0011" =>outMux<= chanel_3;
34             when "0100" =>outMux<= chanel_4;
35             when "0101" =>outMux<= chanel_5;
36             when "0110" =>outMux<= chanel_6;
37             when "0111" =>outMux<= chanel_7;
38             when "1000" =>outMux<= chanel_8;
39             when "1001" =>outMux<= chanel_9;
40             when "1010" =>outMux<= chanel_10;
41             when "1011" =>outMux<= chanel_11;
42             when "1100" =>outMux<= chanel_12;
43             when "1101" =>outMux<= chanel_13;
44             when "1110" =>outMux<= chanel_14;
45             when others => outMux<= chanel_15;
46         end case;
47     end process;
48 end Behavioral;
```

1.2. Mux de 2 canales

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 entity mux2 is
4     Port(p,s: in STD_LOGIC_VECTOR (15 downto 0);
5         sel : in STD_LOGIC;
6         sout : out STD_LOGIC_VECTOR (15 downto 0));
7 end mux2;
8
9 architecture Behavioral of mux2 is
10 begin
11     process(sel)
12     begin
13         if (sel = '0') then
14             sout <= p;
15         else
16             sout <= s;
17         end if;
18     end process;
19 end Behavioral;
```

1.3. Demux de 16 canales

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 entity demuxG is
4     Port(output:out std_logic_vector(15 downto 0);
5         input:in std_logic;
6         selector:in std_logic_vector(3 downto 0));
7 end demuxG;
8
9 architecture Behavioral of demuxG is
10
11 begin
12     process(selector,input)
13     begin
14         if(input = '1') then
15             case selector is
16                 when "0000" =>output<= "0000000000000001";
17                 when "0001" =>output<= "0000000000000010";
18                 when "0010" =>output<= "0000000000000100";
19                 when "0011" =>output<= "0000000000001000";
20                 when "0100" =>output<= "0000000000010000";
21                 when "0101" =>output<= "0000000000100000";
22                 when "0110" =>output<= "0000000001000000";
23                 when "0111" =>output<= "0000000010000000";
24                 when "1000" =>output<= "0000000100000000";
25                 when "1001" =>output<= "0000001000000000";
26                 when "1010" =>output<= "0000010000000000";
27                 when "1011" =>output<= "0000100000000000";
28                 when "1100" =>output<= "0001000000000000";
29                 when "1101" =>output<= "0010000000000000";
30                 when "1110" =>output<= "0100000000000000";
31                 when others => output<= "1000000000000000";
32             end case;
33         else
34             output<="0000000000000000";
35         end if;
36     end process;
37 end Behavioral;
```

1.4. Registro de 16 bits

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
```

```

4 entity registro is
5     Port ( d : in STD_LOGIC_VECTOR (15 downto 0);
6           q : out STD_LOGIC_VECTOR (15 downto 0);
7           clr,clk,l : in STD_LOGIC);
8 end registro;
9
10 architecture Behavioral of registro is
11 begin
12     process(clr,clk,l,d)
13     begin
14         if(clr = '1') then
15             q<=(others=>'0');
16         elsif rising_edge(clk) then
17             if(l='1') then
18                 q<=d;
19             end if;
20         end if;
21     end process;
22 end Behavioral;

```

1.5. Archivo de registros

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4
5 entity archRegistros is
6     Port(writeData : in STD_LOGIC_VECTOR (15 downto 0);
7           writeReg : in STD_LOGIC_VECTOR (3 downto 0);
8           readReg1 : in STD_LOGIC_VECTOR (3 downto 0);
9           readReg2 : in STD_LOGIC_VECTOR (3 downto 0);
10          shamt : in STD_LOGIC_VECTOR (3 downto 0);
11          WR,CLK : in STD_LOGIC;
12          SHE,DIR,clr : in STD_LOGIC;
13          readData1 : inout STD_LOGIC_VECTOR (15 downto 0);
14          readData2 : out STD_LOGIC_VECTOR (15 downto 0));
15 end archRegistros;
16
17 architecture Behavioral of archRegistros is
18     component registro
19     Port(d : in STD_LOGIC_VECTOR (15 downto 0);
20         q : out STD_LOGIC_VECTOR (15 downto 0);
21         clr,clk,l : in STD_LOGIC);
22     end component;
23
24     component barrel
25     Port(dato : in STD_LOGIC_VECTOR (15 downto 0);
26         res : out STD_LOGIC_VECTOR (15 downto 0);
27         dir : std_logic;
28         s : in STD_LOGIC_VECTOR (3 downto 0));
29     end component;
30
31     component muxGral
32     Port (chanel_0:in std_logic_vector(15 downto 0);
33         chanel_1:in std_logic_vector(15 downto 0);
34         chanel_2:in std_logic_vector(15 downto 0);
35         chanel_3:in std_logic_vector(15 downto 0);
36         chanel_4:in std_logic_vector(15 downto 0);
37         chanel_5:in std_logic_vector(15 downto 0);
38         chanel_6:in std_logic_vector(15 downto 0);
39         chanel_7:in std_logic_vector(15 downto 0);
40         chanel_8:in std_logic_vector(15 downto 0);
41         chanel_9:in std_logic_vector(15 downto 0);
42         chanel_10:in std_logic_vector(15 downto 0);
43         chanel_11:in std_logic_vector(15 downto 0);
44         chanel_12:in std_logic_vector(15 downto 0);
45         chanel_13:in std_logic_vector(15 downto 0);

```

```

46         chanel_14:in std_logic_vector(15 downto 0);
47         chanel_15:in std_logic_vector(15 downto 0);
48         selectMux: in STD_LOGIC_VECTOR (3 downto 0);
49         outMux :out std_logic_vector(15 downto 0) );
50     end component;
51
52     component demuxG
53     Port(output:out std_logic_vector(15 downto 0);
54         input:in std_logic;
55         selector:in std_logic_vector(3 downto 0));
56     end component;
57
58     component mux2
59     Port(p,s: in STD_LOGIC_VECTOR (15 downto 0);
60         sel : in STD_LOGIC;
61         sout : out STD_LOGIC_VECTOR (15 downto 0));
62     end component;
63
64     signal BS_out :std_logic_vector(15 downto 0);--salida del barrel shifter
65     signal WRSHE_out :std_logic_vector(15 downto 0);
66     signal WR_writeREG:std_logic_vector(15 downto 0);
67     TYPE banco is array (0 to 15) of std_logic_vector(15 downto 0);
68     signal q:banco;
69
70 begin
71     WR_writeregister: demuxG Port map(
72         output => WR_writeREG,
73         input => WR,
74         selector => writeReg
75     );
76
77     Barrel_shifter:barrel Port map(
78         dato => readData1,
79         res => BS_out,
80         dir => DIR,
81         s => shamt
82     );
83
84     entrada_registros: mux2 port map(
85         sel => SHE,
86         p => writeData,
87         s => BS_out,
88         sout =>WRSHE_out
89     );
90
91     ciclo: for i in 0 to 15 generate
92         Registros:registro PORT MAP(
93             d=>WRSHE_out,
94             q => q(i),
95             clr =>clr,
96             clk => CLK,
97             L =>WR_writeREG(i)
98         );
99     end generate;
100
101     muxreadData1:muxGral Port map(
102         chanel_0 => q(0),
103         chanel_1 => q(1),
104         chanel_2 => q(2),
105         chanel_3 => q(3),
106         chanel_4 => q(4),
107         chanel_5 => q(5),
108         chanel_6 => q(6),
109         chanel_7 => q(7),
110         chanel_8 => q(8),
111         chanel_9 => q(9),
112         chanel_10 => q(10),

```

```

113         chanel_11 => q(11),
114         chanel_12 => q(12),
115         chanel_13 => q(13),
116         chanel_14 => q(14),
117         chanel_15 => q(15),
118         selectMux => readReg1,
119         outMux => readData1
120     );
121     muxreadData2:muxGral Port map(
122         chanel_0 => q(0),
123         chanel_1 => q(1),
124         chanel_2 => q(2),
125         chanel_3 => q(3),
126         chanel_4 => q(4),
127         chanel_5 => q(5),
128         chanel_6 => q(6),
129         chanel_7 => q(7),
130         chanel_8 => q(8),
131         chanel_9 => q(9),
132         chanel_10 => q(10),
133         chanel_11 => q(11),
134         chanel_12 => q(12),
135         chanel_13 => q(13),
136         chanel_14 => q(14),
137         chanel_15 => q(15),
138         selectMux => readReg2,
139         outMux => readData2
140     );
141 end Behavioral;

```

2. Test-Bench VHDL Código

2.1. Test-Bench registro

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity registroTB is
5  -- Port ( );
6  end registroTB;
7
8  architecture Behavioral of registroTB is
9  component registro
10     Port ( d : in STD_LOGIC_VECTOR (15 downto 0);
11           q : out STD_LOGIC_VECTOR (15 downto 0);
12           clr,clk,l : in STD_LOGIC);
13 end component;
14 signal d : STD_LOGIC_VECTOR (15 downto 0);
15 signal q : STD_LOGIC_VECTOR (15 downto 0);
16 signal clr,clk,l : STD_LOGIC;
17 constant periodo : time := 10 ns;
18 begin
19     tb:registro port map(
20         d => d,
21         q => q,
22         clr => clr,
23         clk => clk,
24         l => l
25     );
26     reloj : process
27     begin
28         clk <= '0';
29         wait for 5 ns;
30         clk <= '1';
31         wait for 5 ns;
32     end process;

```

```

33     simulacion:process
34         begin
35             clr <= '1';
36             l <= '0';
37             d <= "0001110011110000";
38             wait for 40 ns;
39             clr <= '0';
40             l <= '1';
41             d <= "0001110011110000";
42             wait for 40 ns;
43             clr <= '0';
44             l <= '0';
45             d <= "0000010011110000";
46             wait for 40 ns;
47             clr <= '0';
48             l <= '1';
49             d <= "1111111111111111";
50             wait for 40 ns;
51             clr <= '1';
52             l <= '0';
53             d <= "0001110011110000";
54             wait for 40 ns;
55             wait;
56         end process;
57     end Behavioral;

```

2.2. Test-Bench barrel shifter

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity barrelTB is
5  -- Port ( );
6  end barrelTB;
7
8  architecture Behavioral of barrelTB is
9  component barrel is
10     Port ( dato : in STD_LOGIC_VECTOR (15 downto 0);
11           res : out STD_LOGIC_VECTOR (15 downto 0);
12           dir : std_logic;
13           s : in STD_LOGIC_VECTOR (3 downto 0));
14  end component;
15
16  signal dato : STD_LOGIC_VECTOR (15 downto 0);
17  signal res : STD_LOGIC_VECTOR (15 downto 0);
18  signal dir: std_logic;
19  signal s : STD_LOGIC_VECTOR (3 downto 0);
20  begin
21     u1 : Barrel
22     Port map (
23         dato => dato,
24         res => res,
25         dir => dir,
26         s => s
27     );
28
29     simulacion : process
30     begin
31         dato <= "0000000010010111";
32         s <= "0011";
33         dir <= '0';
34         wait for 10 ns;
35         dato <= "0000110010010111";
36         s <= "1011";
37         dir <= '0';
38         wait for 10 ns;
39         dato <= "1111111110011111";

```

```

40     s <= "0011";
41     dir <='1';
42     wait for 10 ns;
43     dato <= "1001011110011111";
44     s <= "1011";
45     dir <='1';
46     wait for 10 ns;
47     wait;
48 end process;
49
50
51
52 end Behavioral;

```

2.3. Test-Bench archivo de registros

```

1  LIBRARY ieee;
2  LIBRARY STD;
3  USE STD.TEXTIO.ALL;
4  USE ieee.std_logic_TEXTIO.ALL;    --PERMITE USAR STD_LOGIC
5  USE ieee.std_logic_1164.ALL;
6  USE ieee.std_logic_UNSIGNED.ALL;
7  USE ieee.std_logic_ARITH.ALL;
8
9
10 entity archRegTB is
11 -- Port ( );
12 end archRegTB;
13
14 architecture Behavioral of archRegTB is
15     component archRegistros
16     Port(writeData : in STD_LOGIC_VECTOR (15 downto 0);
17           writeReg : in STD_LOGIC_VECTOR (3 downto 0);
18           readReg1 : in STD_LOGIC_VECTOR (3 downto 0);
19           readReg2 : in STD_LOGIC_VECTOR (3 downto 0);
20           shamt : in STD_LOGIC_VECTOR (3 downto 0);
21           WR,CLK : in STD_LOGIC;
22           SHE,DIR,clr : in STD_LOGIC;
23           readData1 : inout STD_LOGIC_VECTOR (15 downto 0);
24           readData2 : out STD_LOGIC_VECTOR (15 downto 0));
25     end component;
26
27     --inputs
28     signal writeData : STD_LOGIC_VECTOR (15 downto 0);
29     signal writeReg : STD_LOGIC_VECTOR (3 downto 0);
30     signal readReg1 : STD_LOGIC_VECTOR (3 downto 0);
31     signal readReg2 : STD_LOGIC_VECTOR (3 downto 0);
32     signal shamt : STD_LOGIC_VECTOR (3 downto 0);
33     signal WR,CLK : STD_LOGIC := '0';
34     signal SHE,DIR,clr : STD_LOGIC;
35
36     --outputs
37     signal readData1 : STD_LOGIC_VECTOR (15 downto 0);
38     signal readData2 : STD_LOGIC_VECTOR (15 downto 0);
39
40     --clock
41     constant CLK_period : time := 10 ns;
42
43 begin
44     tb_AR:archRegistros port map(
45         writeData =>writeData,
46         writeReg => writeReg,
47         readReg1 => readReg1,
48         readReg2 => readReg2,
49         shamt => shamt,
50         WR => WR,
51         CLK => CLK,

```

```

52     SHE => SHE,
53     DIR => DIR,
54     clr => clr,
55     readData1 => readData1,
56     readData2 => readData2
57 );
58 CLK_process :process
59     begin
60         CLK <= '0';
61         wait for CLK_period/2;
62         CLK <= '1';
63         wait for CLK_period/2;
64     end process;
65     simulacion:process
66         file ARCH_SAL : TEXT;
67         variable LINEA_SAL: line;
68         variable VAR_RD1: STD_LOGIC_VECTOR (15 downto 0);
69         variable VAR_RD2: STD_LOGIC_VECTOR (15 downto 0);
70
71         file ARCH_ENT :TEXT;
72         variable LINEA_ENT :line;
73         variable VAR_WD : STD_LOGIC_VECTOR (15 downto 0);
74         variable VAR_WrRe : STD_LOGIC_VECTOR (3 downto 0);
75         variable VAR_RR1 : STD_LOGIC_VECTOR (3 downto 0);
76         variable VAR_RR2 : STD_LOGIC_VECTOR (3 downto 0);
77         variable VAR_shamt : STD_LOGIC_VECTOR (3 downto 0);
78         variable VAR_WR: STD_LOGIC;
79         variable VAR_SHE,VAR_DIR,VAR_clr : STD_LOGIC;
80
81         VARIABLE CADENA : STRING(1 TO 6);
82
83     begin
84         file_open(ARCH_ENT, "/media/d3vcr4ck/externData/materias-Sem20_2/
arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicavivado/registros/input.
txt", READ_MODE);
85         file_open(ARCH_SAL, "/media/d3vcr4ck/externData/materias-Sem20_2/
arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicavivado/registros/output.
txt", WRITE_MODE);
86
87         CADENA := "_RR1_";
88         write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "RR1"
89         CADENA := " _RR2";
90         write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "RR2"
91         CADENA := " _SHAMT";
92         write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "SHAMT"
93         CADENA := " _WREG";
94         write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "WREG"
95         CADENA := " _WD_";
96         write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "WD"
97         CADENA := " _WR_";
98         write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "WR"
99         CADENA := " _SHE_";
100        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "SHE"
101        CADENA := " _DIR_";
102        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "DIR"
103        CADENA := " _RD1_";
104        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "RD1"
105        CADENA := " _RD2_";
106        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1); --ESCRIBE LA CADENA "RD2"
107        writeline(ARCH_SAL,LINEA_SAL);
108
109
110
111        wait for 100 ns;
112        for I in 0 to 11 loop
113            readline(ARCH_ENT,LINEA_ENT);
114

```



```

115     Hread(LINEA_ENT,VAR_WD);
116     writeData <= VAR_WD;
117     Hread(LINEA_ENT,VAR_WrRe);
118     writeReg <= VAR_WrRe;
119     Hread(LINEA_ENT,VAR_RR1);
120     readReg1 <= VAR_RR1;
121     Hread(LINEA_ENT,VAR_RR2);
122     readReg2 <= VAR_RR2;
123     Hread(LINEA_ENT,VAR_shamt);
124     shamt <= VAR_shamt;
125     read(LINEA_ENT,VAR_WR);
126     WR <= VAR_WR;
127     read(LINEA_ENT,VAR_SHE);
128     SHE <= VAR_SHE;
129     read(LINEA_ENT,VAR_DIR);
130     DIR <= VAR_DIR;
131     read(LINEA_ENT,VAR_clr);
132     clr <= VAR_clr;
133
134     --wait until rising_edge(CLK);
135     wait for 10ns;
136     VAR_RD1 := readData1;
137     VAR_RD2 := readData2;
138
139
140     HWRITE(LINEA_SAL,VAR_RR1, right, 5);
141     HWRITE(LINEA_SAL,VAR_RR2, right, 5);
142     WRITE(LINEA_SAL,VAR_shamt, right, 5);
143     HWRITE(LINEA_SAL,VAR_WrRe, right, 5);
144     HWRITE(LINEA_SAL,VAR_WD, right, 5);
145     WRITE(LINEA_SAL,VAR_WR,right,5);
146     WRITE(LINEA_SAL,VAR_SHE, right, 5);
147     WRITE(LINEA_SAL,VAR_DIR, right, 5);
148     HWRITE(LINEA_SAL,VAR_RD1, right, 5);
149     HWRITE(LINEA_SAL,VAR_RD2, right, 5);
150
151     writeline(ARCH_SAL,LINEA_SAL);
152     end loop;
153     file_close(ARCH_ENT);
154     file_close(ARCH_SAL);
155     wait;
156 end process;
157 end Behavioral;

```

2.3.1. Archivos de entrada y salida para esta sección

Archivo de entrada (input.txt)

```

1 0f86 5 8 4 0 1 0 1 1
2 0089 1 1 0 0 1 0 0 0
3 0072 2 2 1 0 1 0 0 0
4 0123 3 3 0 0 1 0 0 0
5 0053 4 4 0 0 1 0 0 0
6 ffff f 1 2 0 0 0 0 0
7 eeee e 3 4 0 0 0 0 0
8 1234 2 1 2 3 1 1 0 0
9 5555 4 3 4 5 1 1 1 0
10 ffff f 1 2 0 0 0 0 0
11 eeee e 3 4 0 0 0 0 0
12 0f86 5 8 4 0 1 0 1 1
13
14
15
16
17
18 WriteData|writeReg|RR1|RR2|shamt|WR|SHE|DIR|clr

```

Archivo de salida (output.txt)

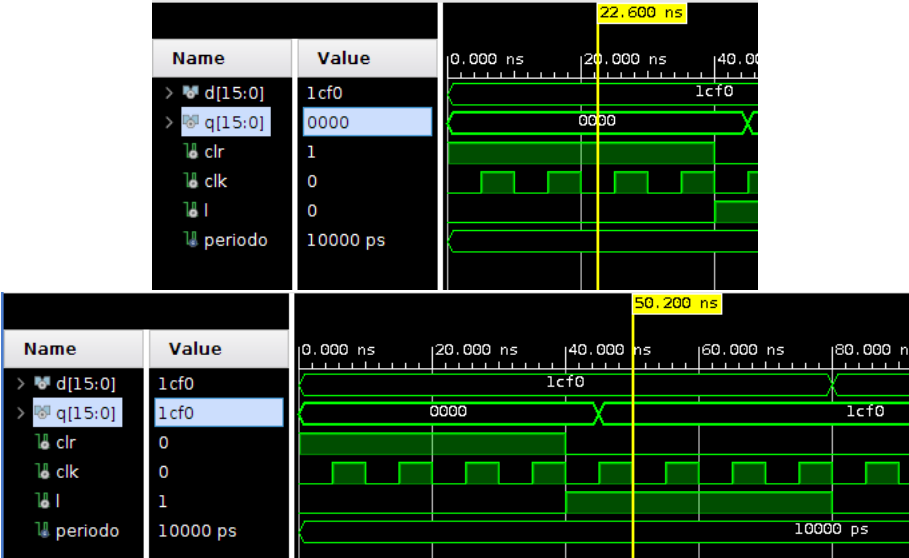
| | _RR1_ | _RR2_ | SHAMT | _WREG | _WD_ | _WR_ | _SHE_ | _DIR_ | _RD1_ | _RD2_ |
|----|-------|-------|-------|--------|------|------|--------|-------|-------|-------|
| 1 | 8 | 4 | 0000 | 5 0F86 | 1 | 0 | 1 0000 | 0000 | | |
| 2 | 1 | 0 | 0000 | 1 0089 | 1 | 0 | 0 0089 | 0000 | | |
| 3 | 2 | 1 | 0000 | 2 0072 | 1 | 0 | 0 0072 | 0089 | | |
| 4 | 3 | 0 | 0000 | 3 0123 | 1 | 0 | 0 0123 | 0000 | | |
| 5 | 4 | 0 | 0000 | 4 0053 | 1 | 0 | 0 0053 | 0000 | | |
| 6 | 1 | 2 | 0000 | F FFFF | 0 | 0 | 0 0089 | 0072 | | |
| 7 | 3 | 4 | 0000 | E EEEE | 0 | 0 | 0 0123 | 0053 | | |
| 8 | 1 | 2 | 0011 | 2 1234 | 1 | 1 | 0 0089 | 0448 | | |
| 9 | 3 | 0 | 0101 | 4 5555 | 1 | 1 | 1 0123 | 0000 | | |
| 10 | 1 | 2 | 0000 | F FFFF | 0 | 0 | 0 0089 | 0448 | | |
| 11 | 3 | 4 | 0000 | E EEEE | 0 | 0 | 0 0123 | 0009 | | |
| 12 | 8 | 4 | 0000 | 5 0F86 | 1 | 0 | 1 0000 | 0000 | | |
| 13 | | | | | | | | | | |

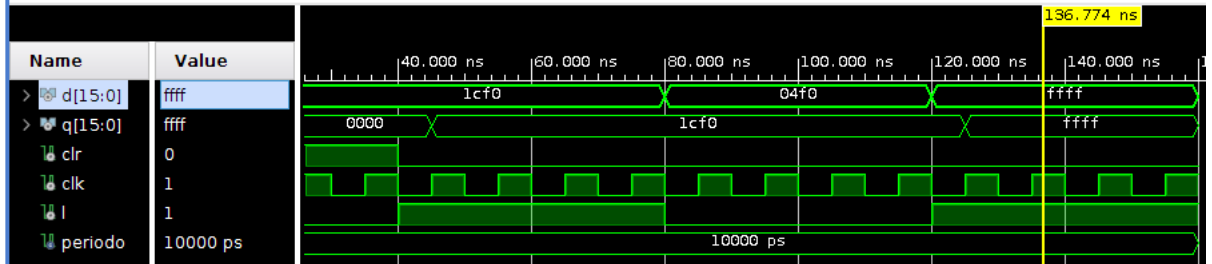
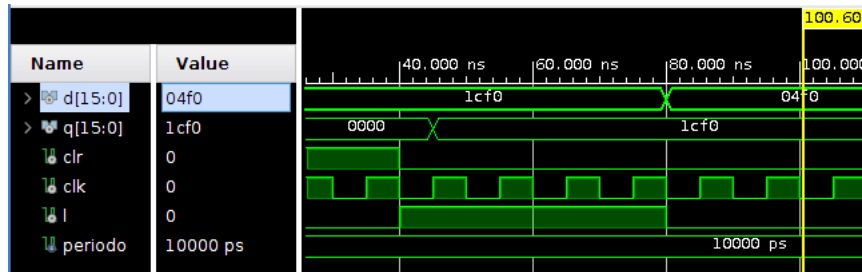
2.3.2. Tabla de resultados de la salida

| RR1 | RR2 | SHAMT | WREG | WD | WR | SHE | DIR | RD1 | RD2 |
|-----|-----|-------|------|------|----|-----|-----|------|------|
| 8 | 4 | 0000 | 5 | 0F86 | 1 | 0 | 1 | 0000 | 0000 |
| 1 | 0 | 0000 | 1 | 0089 | 1 | 0 | 0 | 0089 | 0000 |
| 2 | 1 | 0000 | 2 | 0072 | 1 | 0 | 0 | 0072 | 0089 |
| 3 | 0 | 0000 | 3 | 0123 | 1 | 0 | 0 | 0123 | 0000 |
| 4 | 0 | 0000 | 4 | 0053 | 1 | 0 | 0 | 0053 | 0000 |
| 1 | 2 | 0000 | F | FFFF | 0 | 0 | 0 | 0089 | 0072 |
| 3 | 4 | 0000 | E | EEEE | 0 | 0 | 0 | 0123 | 0053 |
| 1 | 2 | 0011 | 2 | 1234 | 1 | 1 | 0 | 0089 | 0448 |
| 3 | 0 | 0101 | 4 | 5555 | 1 | 1 | 1 | 0123 | 0000 |
| 1 | 2 | 0000 | F | FFFF | 0 | 0 | 0 | 0089 | 0448 |
| 3 | 4 | 0000 | E | EEEE | 0 | 0 | 0 | 0123 | 0009 |
| 8 | 4 | 0000 | 5 | 0F86 | 1 | 0 | 1 | 0000 | 0000 |

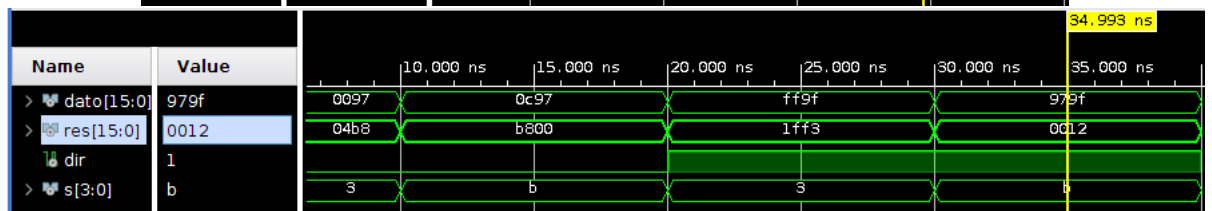
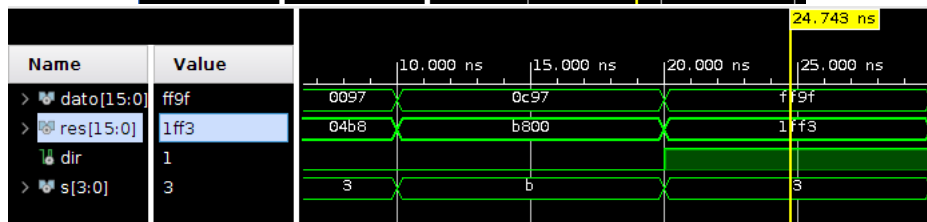
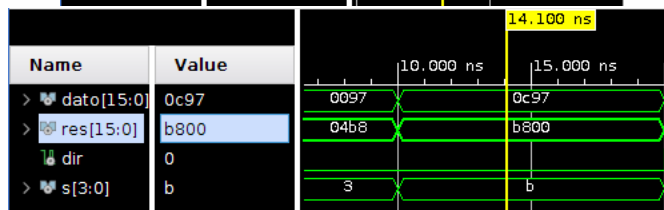
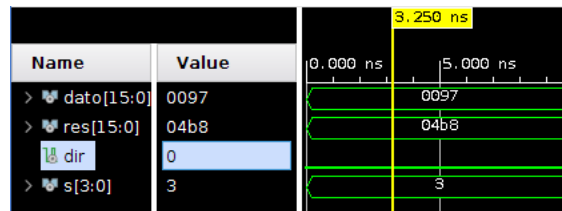
3. Simulaciones

3.1. Registro



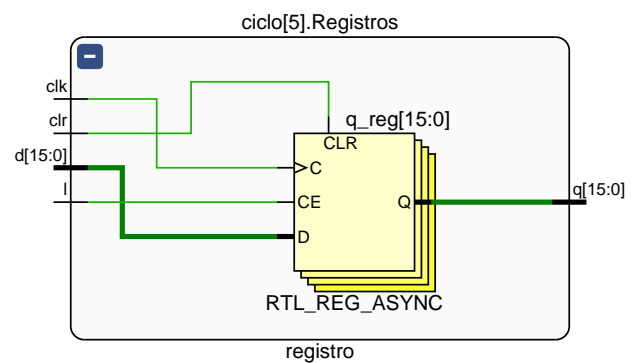


3.2. Barrel-Shifter

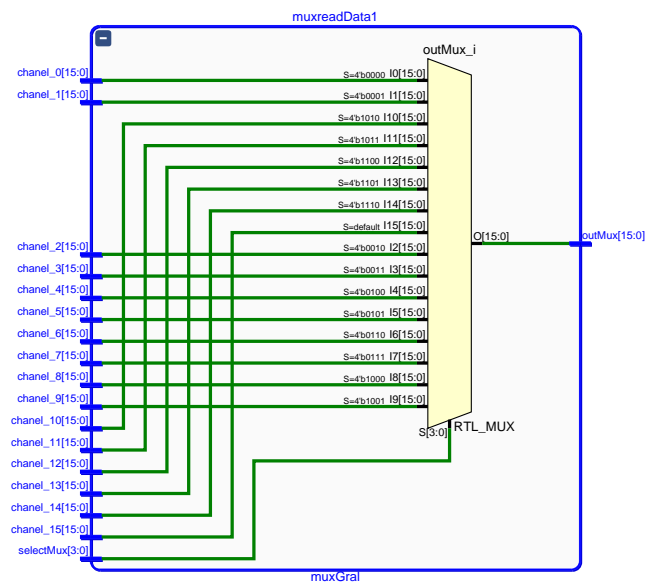


4. Diagramas RTL

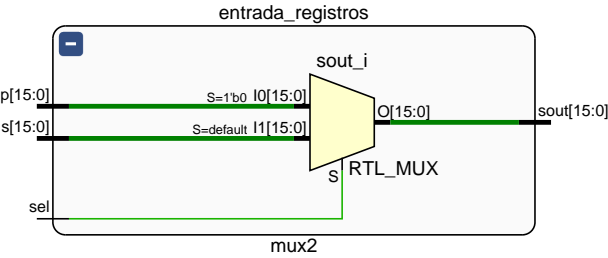
4.1. Registro



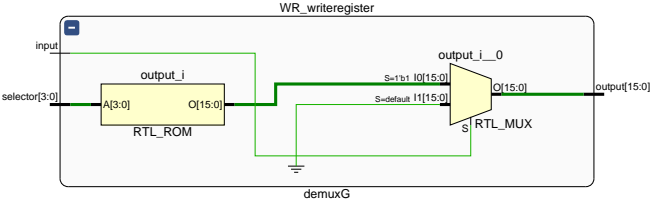
4.2. Mux 16 canales



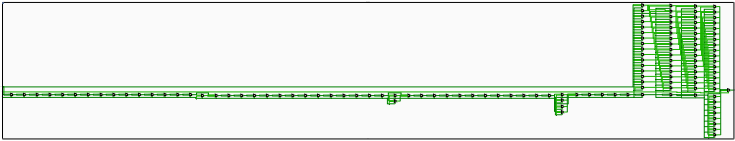
4.3. Mux 2 canales



4.4. Demux



4.5. Barrel-Shifter



4.6. Archivo de Registros

