Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 6 - Archivo de Registros

Unidad de aprendizaje: Arquitectura de Computadoras

Grupo: 3CV1

Alumno(a):
Ramos Diaz Enrique

Profesor(a):
Vega García Nayeli

1 de abril 2020

Índice

1. Código de implementación

1.1. Registro

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

entity Registro is
    generic ( n : integer := 16 );
    Port ( d : in STD_LOGIC_VECTOR (n-1 downto 0);
           q : out STD_LOGIC_VECTOR (n-1 downto 0);
           clr, clk, l : in STD_LOGIC);
end Registro;

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

1.2. Multiplexor de 16 canales

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

entity Multiplexor is
    Port ( cl0, cl1, cl2, cl3, cl4, cl5, cl6, cl7, cl8, cl9, cl10, cl11,
     ↪  cl12, cl13, cl14, cl15 : in STD_LOGIC_VECTOR (15 downto 0);
           sel : in STD_LOGIC_VECTOR (3 downto 0);
           salida : out STD_LOGIC_VECTOR (15 downto 0));
end Multiplexor;

architecture Behavioral of Multiplexor is
begin
    with sel select salida <=
```

```
        cl0 when "0000",
        cl1 when "0001",
        cl2 when "0010",
        cl3 when "0011",
        cl4 when "0100",
        cl5 when "0101",
        cl6 when "0110",
        cl7 when "0111",
        cl8 when "1000",
        cl9 when "1001",
        cl10 when "1010",
        cl11 when "1011",
        cl12 when "1100",
        cl13 when "1101",
        cl14 when "1110",
        cl15 when others;
end Behavioral;
```

1.3. Demultiplexor de 16 canales de salida

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

entity Demultiplexor is
    Port ( d : in STD_LOGIC;
           l : out std_logic_vector(15 downto 0);
           sel : in STD_LOGIC_VECTOR (3 downto 0));
end Demultiplexor;

architecture Behavioral of Demultiplexor is
begin
    process(d, sel)
    begin
        case sel is
            when "0000" =>
                l <= (others => '0');
                l(0) <= d;
            when "0001" =>
                l <= (others => '0');
                l(1) <= d;
            when "0010" =>
                l <= (others => '0');
                l(2) <= d;
```

```
24              when "0011" =>
25                  l <= (others => '0');
26                  l(3) <= d;
27              when "0100" =>
28                  l <= (others => '0');
29                  l(4) <= d;
30              when "0101" =>
31                  l <= (others => '0');
32                  l(5) <= d;
33              when "0110" =>
34                  l <= (others => '0');
35                  l(6) <= d;
36              when "0111" =>
37                  l <= (others => '0');
38                  l(7) <= d;
39              when "1000" =>
40                  l <= (others => '0');
41                  l(8) <= d;
42              when "1001" =>
43                  l <= (others => '0');
44                  l(9) <= d;
45              when "1010" =>
46                  l <= (others => '0');
47                  l(10) <= d;
48              when "1011" =>
49                  l <= (others => '0');
50                  l(11) <= d;
51              when "1100" =>
52                  l <= (others =>'0');
53                  l(12) <= d;
54              when "1101" =>
55                  l <= (others => '0');
56                  l(13) <= d;
57              when "1110" =>
58                  l <= (others => '0');
59                  l(14) <= d;
60              when others =>
61                  l <= (others => '0');
62                  l(15) <= d;
63          end case;
64      end process;
65  end Behavioral;
```

1.4. Barrel-Shifter

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

entity BarrelShifter is
    generic ( n : integer := 16;
              corr : integer := 4);
    Port ( dato : in STD_LOGIC_VECTOR (n-1 downto 0);
           res : out STD_LOGIC_VECTOR (n-1 downto 0);
           shamt : in STD_LOGIC_VECTOR (corr-1 downto 0);
           direccion : in STD_LOGIC);
end BarrelShifter;

architecture Behavioral of BarrelShifter is
begin
    process(dato, shamt, direccion)
        variable aux : std_logic_vector(n-1 downto 0);
    begin
        aux := dato;
        for i in 0 to corr-1 loop
            if (shamt(i) = '0') then
                aux := aux;
            else
                case direccion is
                    when '1' => --- izquierda
                        for j in n-1 downto 0 loop          --el ciclo for
                         ↪ para el corrimiento a la izquierda
                            if (j < 2**i) then              --va del mas
                             ↪ significativo al menos significativo
                                aux(j) := '0';              --para que los
                                 ↪ bits se vayan arrastrando y que los
                                 ↪ ultimos
                            else                            --en
                             ↪ actualizarse sean los de la derecha que es
                             ↪ en donde se ingrean
                                aux(j) := aux(j-2**i);      --los 0s por el
                                 ↪ corrimiento, si lo recorremos a la
                                 ↪ inversa, se actualizaria
                            end if;                         --primero el bit
                             ↪ menos significativo y ese valor se
                             ↪ replicaria en todo el vector
                        end loop;                           --teniendo como
                         ↪ resultado un vector lleno de 0s
```

```
                    when others => --- derecha
                        for j in 0 to n-1 loop              -- invirtiendo
                        ↪  el for
                            if (j <= (n-1) - 2**i) then
                                aux(j) := aux(j+2**i);
                            else
                                aux(j) := '0';
                            end if;
                        end loop;
                end case;
            end if;
        end loop;
        res <= aux;
    end process;
end Behavioral;
```

1.5. Multiplexor de 2 canales

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

entity Mux2C is
    generic ( n : integer := 16 );
    Port ( barrelShifter, write_data : in STD_LOGIC_VECTOR (n-1 downto 0);
           she : in STD_LOGIC;
           sal : out STD_LOGIC_VECTOR (n-1 downto 0));
end Mux2C;

architecture Behavioral of Mux2C is
begin
    with she select sal <=
        barrelShifter when '1',
        write_data when others;
end Behavioral;
```

1.6. Archivo de Registros

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

entity ArchivoRegistros is
    generic ( i : integer := 4;
              n : integer := 16);
    Port ( wr, dir, she, clk, clr : in STD_LOGIC;
           writeReg, readReg1, readReg2, shamt : in STD_LOGIC_VECTOR (i-1
            ↪ downto 0);
           writeData : in STD_LOGIC_VECTOR (n-1 downto 0);
           readData1, readData2 : out STD_LOGIC_VECTOR (n-1 downto 0));
end ArchivoRegistros;

architecture Behavioral of ArchivoRegistros is
    component Registro is
        Port ( d : in STD_LOGIC_VECTOR (n-1 downto 0);
               q : out STD_LOGIC_VECTOR (n-1 downto 0);
               clr, clk, l : in STD_LOGIC);
    end component;

    component Demultiplexor is
        Port ( d : in STD_LOGIC;
               l : out std_logic_vector(15 downto 0);
               sel : in STD_LOGIC_VECTOR (3 downto 0));
    end component;

    component BarrelShifter is
        Port ( dato : in STD_LOGIC_VECTOR (n-1 downto 0);
               res : out STD_LOGIC_VECTOR (n-1 downto 0);
               shamt : in STD_LOGIC_VECTOR (i-1 downto 0);
               direccion : in STD_LOGIC);
    end component;

    component Multiplexor is
        Port ( cl0, cl1, cl2, cl3, cl4, cl5, cl6, cl7, cl8, cl9, cl10, cl11,
         ↪ cl12, cl13, cl14, cl15 : in STD_LOGIC_VECTOR (15 downto 0);
               sel : in STD_LOGIC_VECTOR (3 downto 0);
               salida : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    component Mux2C is
```

```
40          Port ( barrelShifter, write_data : in STD_LOGIC_VECTOR (n-1 downto
            ↪ 0);
41                 she : in STD_LOGIC;
42                 sal : out STD_LOGIC_VECTOR (n-1 downto 0));
43      end component;
44
45      signal auxReadData1, auxL, auxD, auxBS: STD_LOGIC_VECTOR (n-1 downto 0);
46      type matrizQ is array (0 to n-1) of STD_LOGIC_VECTOR (n-1 downto 0);
47      signal auxQ : matrizQ;
48  begin
49      demux : Demultiplexor Port map (
50          d => wr,
51          l => auxL,
52          sel => writeReg
53      );
54
55      registros: for r in 0 to n-1 generate
56          reg: Registro Port map (
57             d => auxD,
58             q => auxQ(r),
59             clk => clk,
60             clr => clr,
61             l => auxL(r)
62          );
63      end generate;
64
65      mux1 : Multiplexor Port map (
66          cl0 => auxQ(0),
67          cl1 => auxQ(1),
68          cl2 => auxQ(2),
69          cl3 => auxQ(3),
70          cl4 => auxQ(4),
71          cl5 => auxQ(5),
72          cl6 => auxQ(6),
73          cl7 => auxQ(7),
74          cl8 => auxQ(8),
75          cl9 => auxQ(9),
76          cl10 => auxQ(10),
77          cl11 => auxQ(11),
78          cl12 => auxQ(12),
79          cl13 => auxQ(13),
80          cl14 => auxQ(14),
81          cl15 => auxQ(15),
82          sel => readReg1,
```

```
83          salida => auxReadData1
84      );
85      readData1 <= auxReadData1;
86
87      mux2 : Multiplexor Port map (
88          cl0 => auxQ(0),
89          cl1 => auxQ(1),
90          cl2 => auxQ(2),
91          cl3 => auxQ(3),
92          cl4 => auxQ(4),
93          cl5 => auxQ(5),
94          cl6 => auxQ(6),
95          cl7 => auxQ(7),
96          cl8 => auxQ(8),
97          cl9 => auxQ(9),
98          cl10 => auxQ(10),
99          cl11 => auxQ(11),
100         cl12 => auxQ(12),
101         cl13 => auxQ(13),
102         cl14 => auxQ(14),
103         cl15 => auxQ(15),
104         sel => readReg2,
105         salida => readData2
106     );
107
108     bs: BarrelShifter Port map(
109         dato => auxReadData1,
110         res => auxBS,
111         shamt => shamt,
112         direccion => dir
113     );
114
115     mux_2C: Mux2C Port map (
116         barrelShifter => auxBs,
117         write_data => writeData,
118         she => she,
119         sal => auxD
120     );
121 end Behavioral;
```

2. Código de simulación

2.1. Registro

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

entity tbRegistro is
end tbRegistro;

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

    signal d : STD_LOGIC_VECTOR (15 downto 0);
    signal q : STD_LOGIC_VECTOR (15 downto 0);
    signal clr, clk, l : STD_LOGIC;
begin
    reg : Registro Port map (
        d => d,
        q => q,
        clk => clk,
        clr => clr,
        l => l
    );

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

    sim : process begin
        clr <= '1';
        wait for 40 ns;
        clr <= '0';
        l <= '1';
        d <= x"1234";
        wait for 10 ns;
        l <= '0';
```

```
        d <= x"8765";
        wait;
    end process;
end Behavioral;
```

2.2. Demultiplexor de 16 canales de salida

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

entity tbDemultiplexor is
end tbDemultiplexor;

architecture Behavioral of tbDemultiplexor is
    component Demultiplexor is
        Port ( d : in STD_LOGIC;
               l : out std_logic_vector(15 downto 0);
               sel : in STD_LOGIC_VECTOR (3 downto 0));
    end component;

    signal d : STD_LOGIC;
    signal l : std_logic_vector(15 downto 0);
    signal sel : STD_LOGIC_VECTOR (3 downto 0);
begin
    demux: Demultiplexor Port map (
        d => d,
        l => l,
        sel => sel
    );

    process begin
        d <= '1';
        sel <= "0000";
        wait for 62 ns;

        sel <= "0001";
        wait for 62 ns;

        sel <= "0010";
        wait for 62 ns;

        sel <= "0011";
        wait for 62 ns;
```

```
37
38          sel <= "0100";
39          wait for 62 ns;
40
41          sel <= "0101";
42          wait for 62 ns;
43
44          sel <= "0110";
45          wait for 62 ns;
46
47          sel <= "0111";
48          wait for 62 ns;
49
50          sel <= "1000";
51          wait for 62 ns;
52
53          sel <= "1001";
54          wait for 62 ns;
55
56          sel <= "1010";
57          wait for 62 ns;
58
59          sel <= "1011";
60          wait for 62 ns;
61
62          sel <= "1100";
63          wait for 62 ns;
64
65          sel <= "1101";
66          wait for 62 ns;
67
68          sel <= "1110";
69          wait for 62 ns;
70
71          sel <= "1111";
72          wait;
73      end process;
74  end Behavioral;
```

2.3. Barrel-Shifter

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity tbBarrelShifter is
5  end tbBarrelShifter;
6
7  architecture Behavioral of tbBarrelShifter is
8      component BarrelShifter is
9          Port ( dato : in STD_LOGIC_VECTOR (15 downto 0);
10                res : out STD_LOGIC_VECTOR (15 downto 0);
11                shamt : in STD_LOGIC_VECTOR (3 downto 0);
12                direccion : in STD_LOGIC);
13     end component;
14
15     signal dato : STD_LOGIC_VECTOR (15 downto 0);
16     signal res : STD_LOGIC_VECTOR (15 downto 0);
17     signal shamt : STD_LOGIC_VECTOR (3 downto 0);
18     signal direccion : STD_LOGIC;
19 begin
20     barrel : BarrelShifter Port map (
21         dato => dato,
22         res  => res,
23         shamt => shamt,
24         direccion => direccion
25     );
26
27     process begin
28         direccion <= '1'; --- izquierda
29         dato <= "0000000010010111";
30         shamt <= "0011";
31         wait for 500 ns;
32         direccion <= '0'; --- derecha
33         dato <= "0000000010010001";
34         shamt <= "0100";
35         wait;
36     end process;
37 end Behavioral;
```

2.4. Archivo de Registros

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use STD.textio.all;
use IEEE.std_logic_textio.all;

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is
    component ArchivoRegistros is
        Port ( wr, dir, she, clk, clr : in STD_LOGIC;
               writeReg, readReg1, readReg2, shamt : in STD_LOGIC_VECTOR (3
                ↪  downto 0);
               writeData : in STD_LOGIC_VECTOR (15 downto 0);
               readData1, readData2 : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    signal wr, dir, she, clk, clr : STD_LOGIC;
    signal writeReg, readReg1, readReg2, shamt : STD_LOGIC_VECTOR (3 downto
     ↪  0);
    signal writeData, readData1, readData2 : STD_LOGIC_VECTOR (15 downto 0);
begin
    ar: ArchivoRegistros Port map (
        wr => wr,
        she => she,
        dir => dir,
        clk => clk,
        clr => clr,
        writeReg => writeReg,
        readReg1 => readReg1,
        readReg2 => readReg2,
        shamt => shamt,
        writeData => writeData,
        readData1 => readData1,
        readData2 => readData2
    );

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

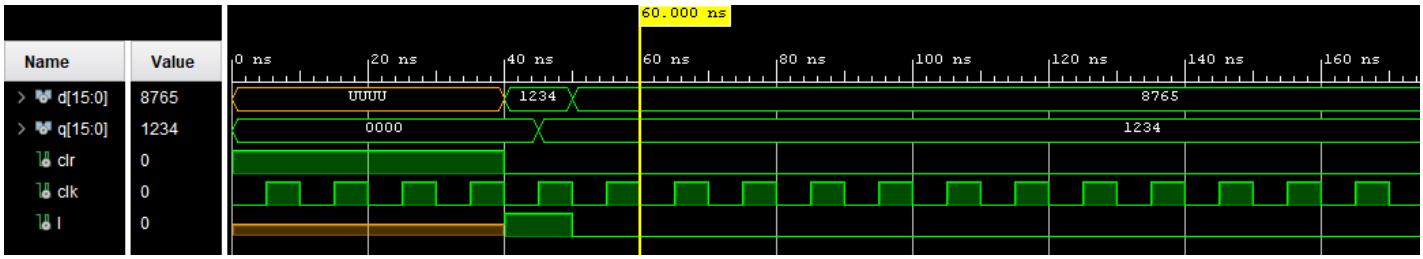
```
41          wait for 5 ns;
42      end process;
43
44      process
45          file arch_res : text;   --Apuntadores tipo
            ↪ txt
46          variable linea_res : line;
47          variable var_read_data1 : STD_LOGIC_VECTOR (15 downto 0);
48          variable var_read_data2 : STD_LOGIC_VECTOR (15 downto 0);
49
50          file arch_en : text; --Apuntadores tipo txt
51          variable linea_en: line;
52          variable var_write : STD_LOGIC_VECTOR (15 downto 0);
53          variable var_wr, var_she, var_dir, var_clr : STD_LOGIC;
54          variable var_write_reg, var_read_reg1, var_read_reg2, var_shamt :
            ↪ STD_LOGIC_VECTOR (3 downto 0);
55          variable var_write_data : STD_LOGIC_VECTOR (15 downto 0);
56          variable cadena : string (1 to 5);
57      begin
58          --- RR1 RR2 SHAMT WREG WD CLR WR SHE DIR
59          file_open(arch_en, "Estimulos.txt", READ_MODE);
60
61          --- RR1 RR2 SHAMT WREG WD WR SHE DIR RD1 RD2
62          file_open(arch_res, "Resultado.txt", WRITE_MODE);
63
64          cadena := "  RR1";
65          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪ "readReg1"
66          cadena := "  RR2";
67          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪ "readReg2"
68          cadena := "SHAMT";
69          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪ "shamt"
70          cadena := " WREG";
71          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪ "writeReg"
72          cadena := "   WD";
73          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪ "writeData"
74          cadena := "   WR";
75          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
            ↪ "WR"
76          cadena := "  SHE";
```

```
77          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
             ↪  "SHE"
78          cadena := "  DIR";
79          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
             ↪  "DIR"
80          cadena := "  RD1";
81          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
             ↪  "readData1"
82          cadena := "  RD2";
83          write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
             ↪  "readData2"
84
85          writeline(arch_res, linea_res);-- escribe la linea en el archivo
86
87          wait for 100 ns;
88          for i in 0 to 11 loop
89              readline(arch_en, linea_en); -- lee una linea completa
90              --- RR1 RR2 SHAMT WREG WD CLR WR SHE DIR
91
92              --Lee readReg1
93              read(linea_en, var_read_reg1);
94              readReg1 <= var_read_reg1;
95
96              --Lee readReg2
97              read(linea_en, var_read_reg2);
98              readReg2 <= var_read_reg2;
99
100             --Lee shamt
101             read(linea_en, var_shamt);
102             shamt <= var_shamt;
103
104             --Lee writeReg
105             read(linea_en, var_write_reg);
106             writeReg <= var_write_reg;
107
108             --Lee writeData
109             read(linea_en, var_write_data);
110             writeData <= var_write_data;
111
112             --Lee clr
113             read(linea_en, var_clr);
114             clr <= var_clr;
115
116             --Lee wr
```

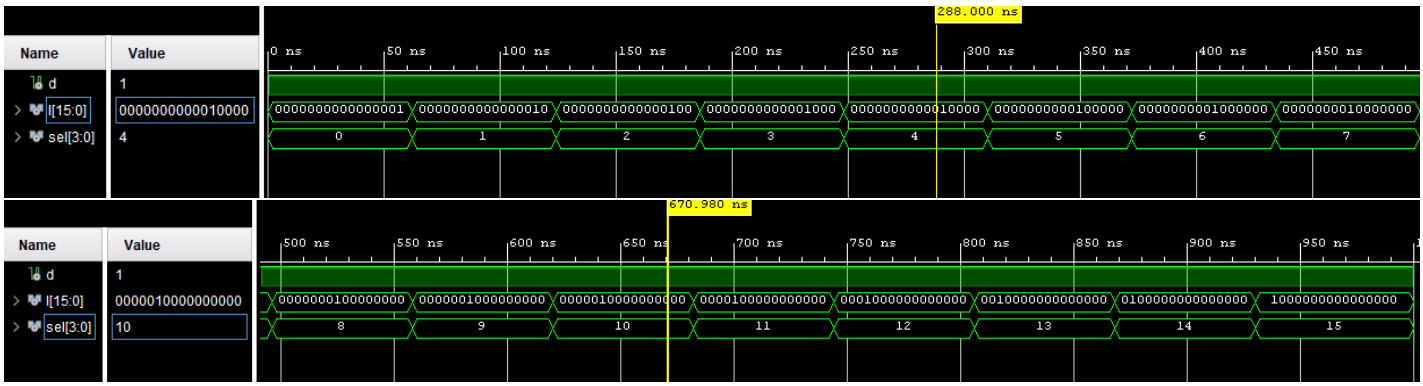
```
117             read(linea_en, var_wr);
118             wr <= var_wr;
119 
120             --Lee she
121             read(linea_en, var_she);
122             she <= var_she;
123 
124             --Lee dir
125             read(linea_en, var_dir);
126             dir <= var_dir;
127 
128             wait until rising_edge(clk); --ESPERA AL FLANCO DE SUBIDA
129             var_read_data1 := readData1;
130             var_read_data2 := readData2;
131             --- RR1 RR2 SHAMT WREG WD WR SHE DIR RD1 RD2
132             Hwrite(linea_res, var_read_reg1, right, 5);--ESCRIBE EL CAMPO
                 ↪ RR1
133             Hwrite(linea_res, var_read_reg2, right, 6);--ESCRIBE EL CAMPO
                 ↪ RR2
134             Hwrite(linea_res, var_shamt, right, 6);--ESCRIBE EL CAMPO shamt
135             Hwrite(linea_res, var_write_reg, right, 5);--ESCRIBE EL CAMPO
                 ↪ WREG
136             Hwrite(linea_res, var_write_data, right, 8);--ESCRIBE EL CAMPO
                 ↪ WD
137             write(linea_res, var_wr, right, 7);--ESCRIBE EL CAMPO WR
138             write(linea_res, var_she, right, 5);--ESCRIBE EL CAMPO SHE
139             write(linea_res, var_dir, right, 6);--ESCRIBE EL CAMPO DIR
140             Hwrite(linea_res, var_read_data1, right, 6);--ESCRIBE EL CAMPO
                 ↪ RD1
141             Hwrite(linea_res, var_read_data2, right, 6);--ESCRIBE EL CAMPO
                 ↪ RD2
142 
143             writeline(arch_res, linea_res);-- escribe la linea en el archivo
144         end loop;
145         file_close(arch_en);  -- cierra el archivo
146         file_close(arch_res);  -- cierra el archivo
147       wait;
148     end process;
149 end Behavioral;
```

3. Simulación

3.1. Registro

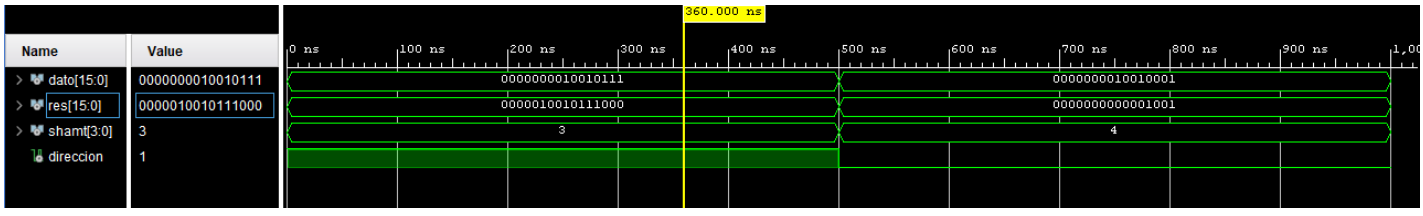


3.2. Demultiplexor de 16 canales de salida

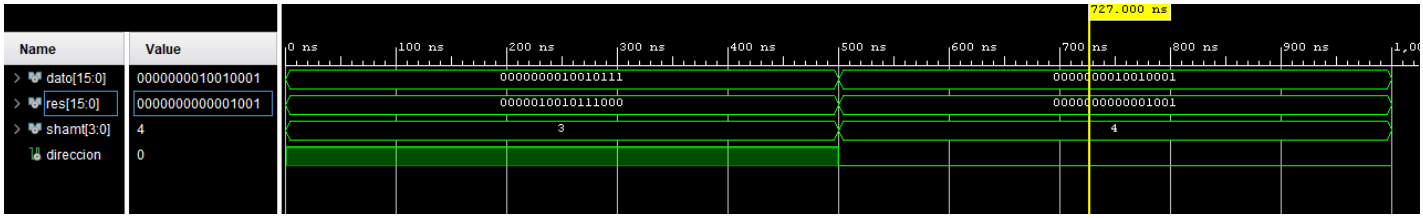


3.3. Barrel-Shifter

3.3.1. Corrimiento a la izquierda

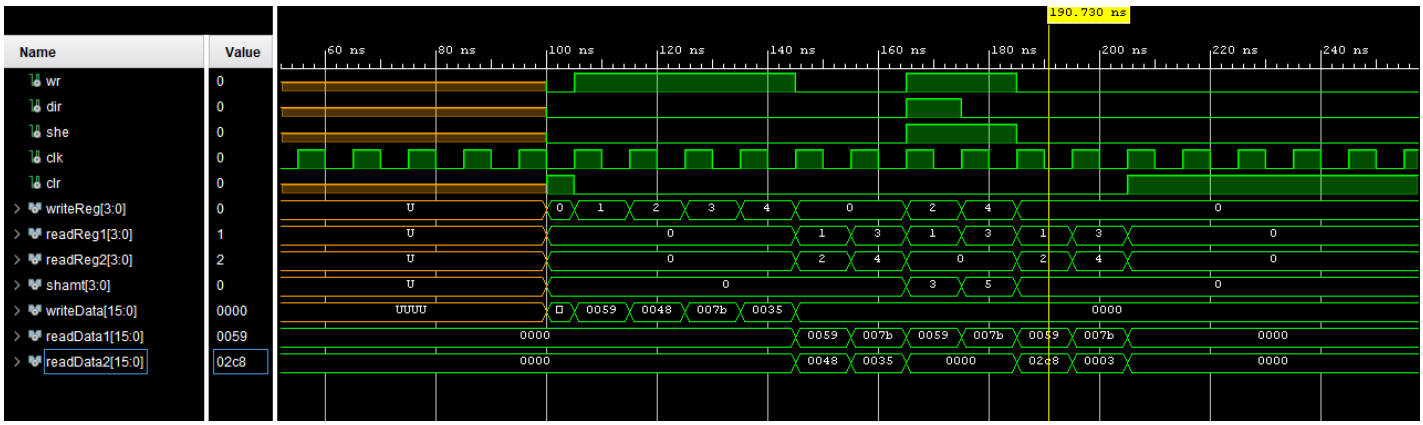


3.3.2. Corrimiento a la derecha



3.4. Archivo de Registros

1. Reset
2. Banco[1] = 89
3. Banco[2] = 72
4. Banco[3] = 123
5. Banco[4] = 53
6. Leer Banco[1] y Banco[2]
7. Leer Banco[3] y Banco[4]
8. Banco[2] = Banco[1] « 3
9. Banco[4] = Banco[3] » 5
10. Leer Banco[1] y Banco[2]
11. Leer Banco[3] y Banco[4]
12. Reset



3.4.1. Archivo entrada: Estimulos.txt

```
--- RR1 RR2 SHAMT WREG WD CLR WR SHE DIR
0000 0000 0000 0000 0000000000000000 1 0 0 0
0000 0000 0000 0001 0000000001011001 0 1 0 0
0000 0000 0000 0010 0000000001001000 0 1 0 0
0000 0000 0000 0011 0000000001111011 0 1 0 0
0000 0000 0000 0100 0000000000110101 0 1 0 0
0001 0010 0000 0000 0000000000000000 0 0 0 0
0011 0100 0000 0000 0000000000000000 0 0 0 0
0001 0000 0011 0010 0000000000000000 0 1 1 1
0011 0000 0101 0100 0000000000000000 0 1 1 0
0001 0010 0000 0000 0000000000000000 0 0 0 0
0011 0100 0000 0000 0000000000000000 0 0 0 0
0000 0000 0000 0000 0000000000000000 1 0 0 0
```

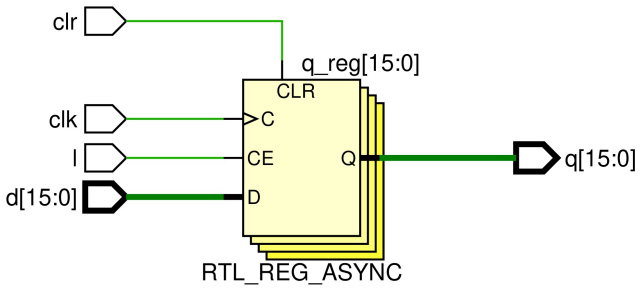
3.4.2. Archivo salida: Resultados.txt

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

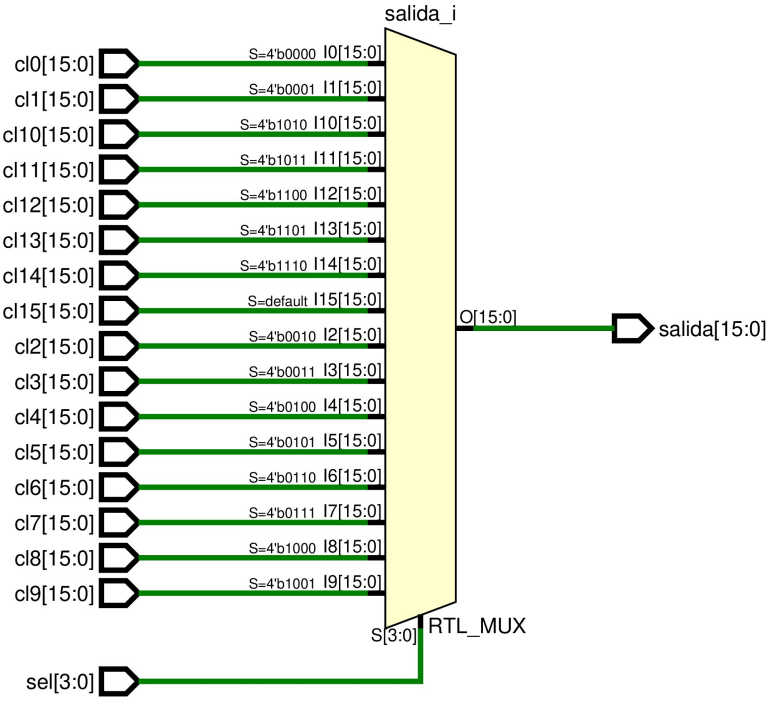
4. Diagramas RTL

4.1. Análisis RTL

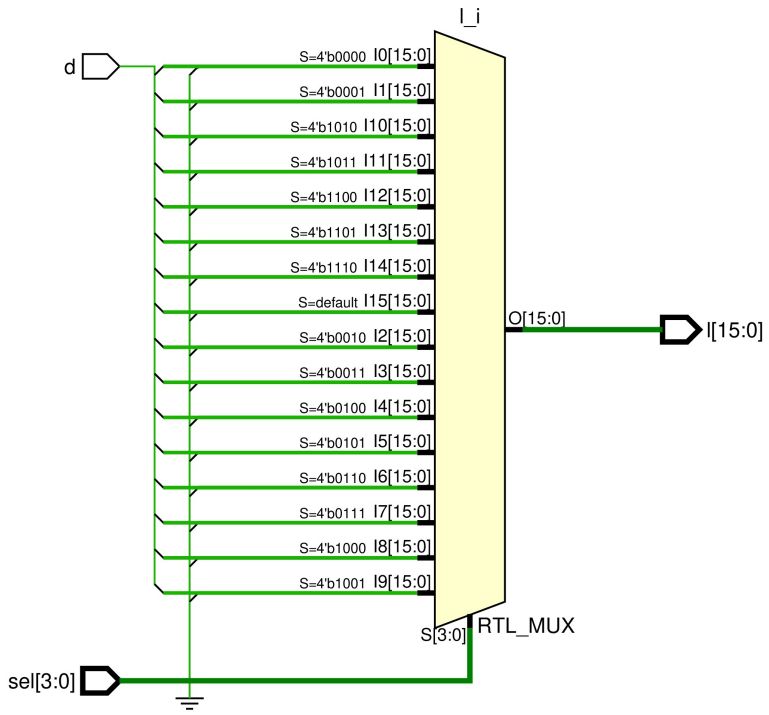
4.1.1. Registro



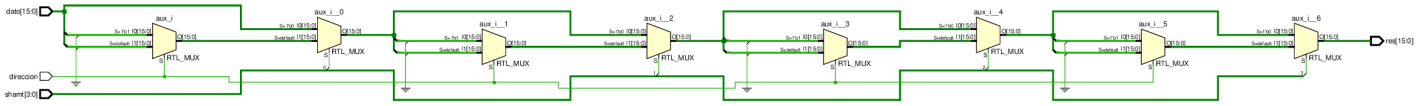
4.1.2. Multiplexor de 16 canales



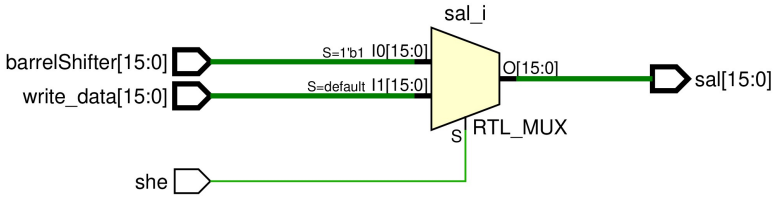
4.1.3. Demultiplexor de 16 canales de salida



4.1.4. Barrel-Shifter



4.1.5. Multiplexor de 2 canales



4.1.6. Archivo de Registros

Diagrama comprimido

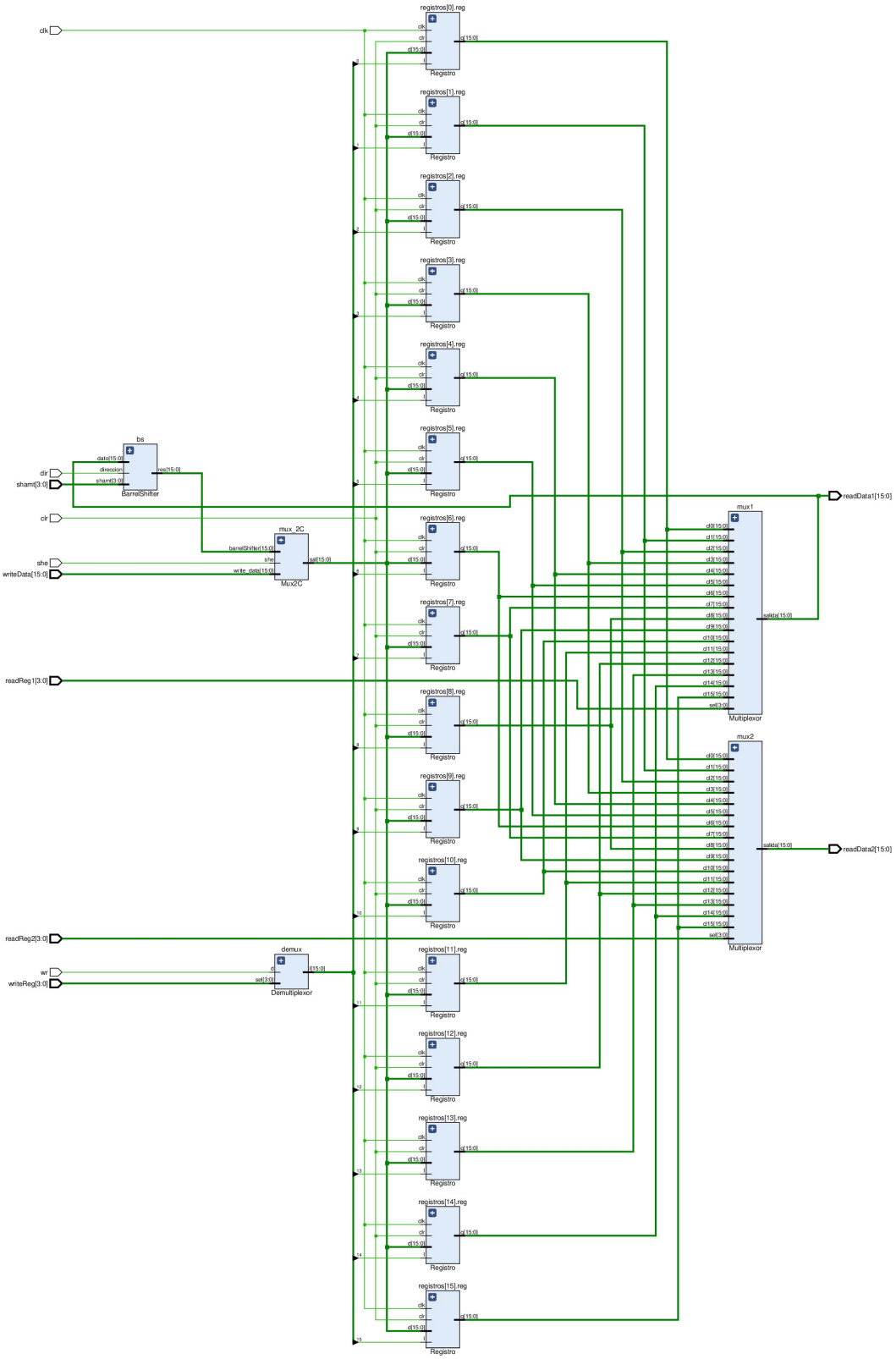
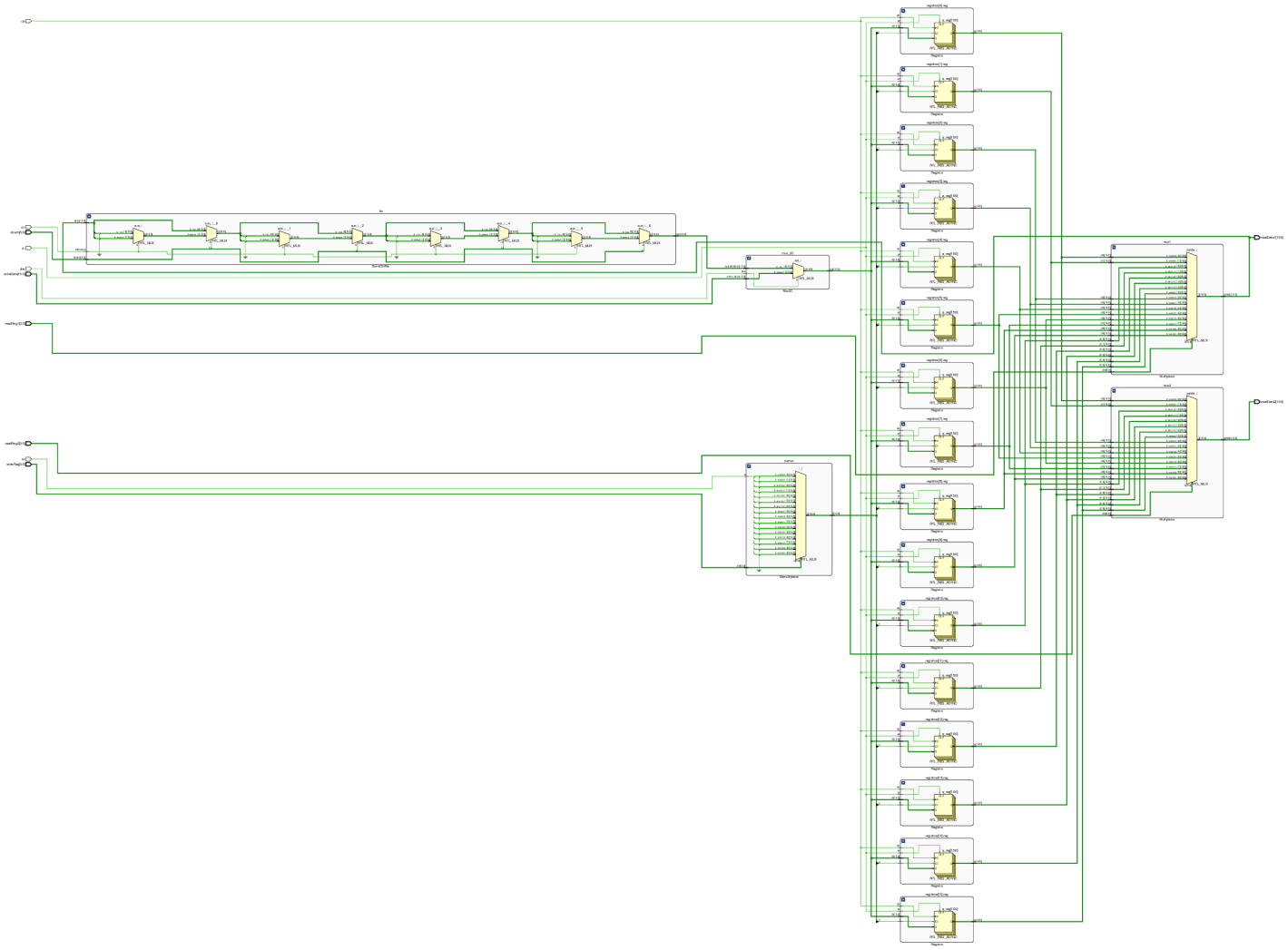
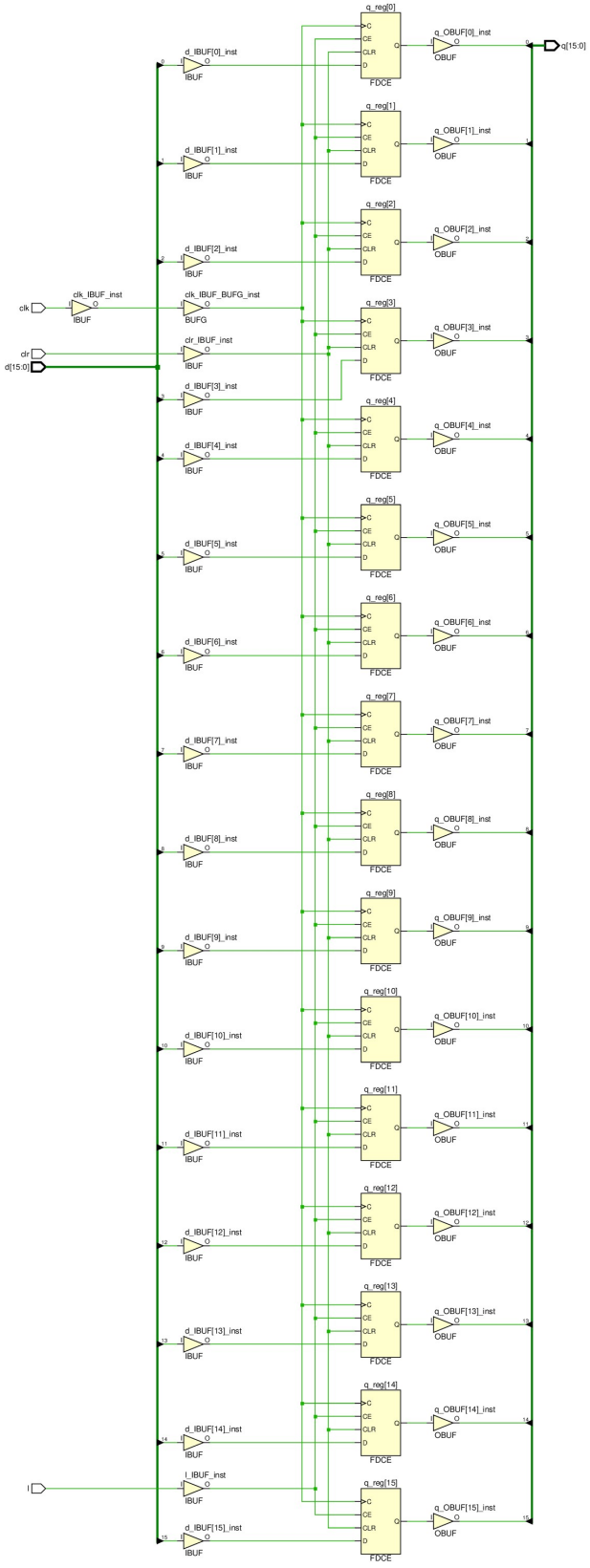


Diagrama expandido

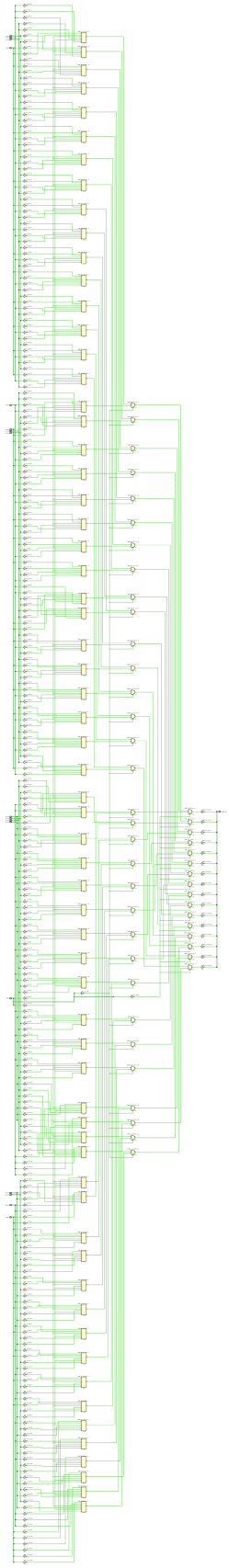


4.2. Synthesis

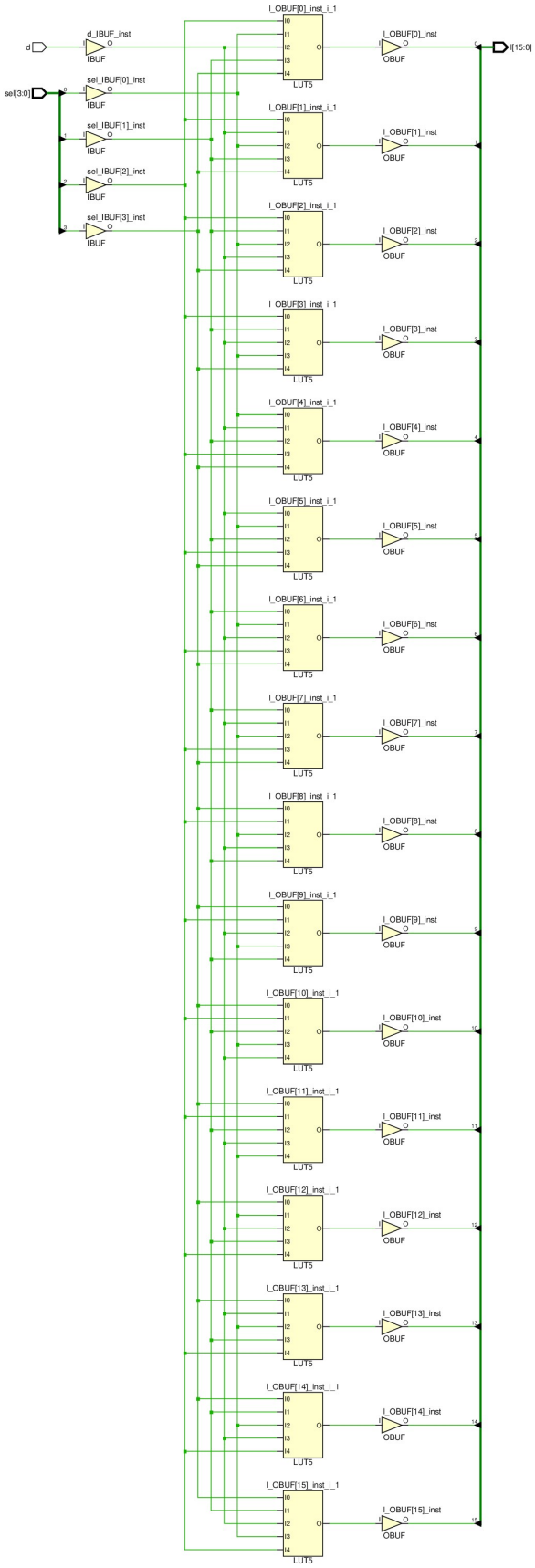
4.2.1. Registro



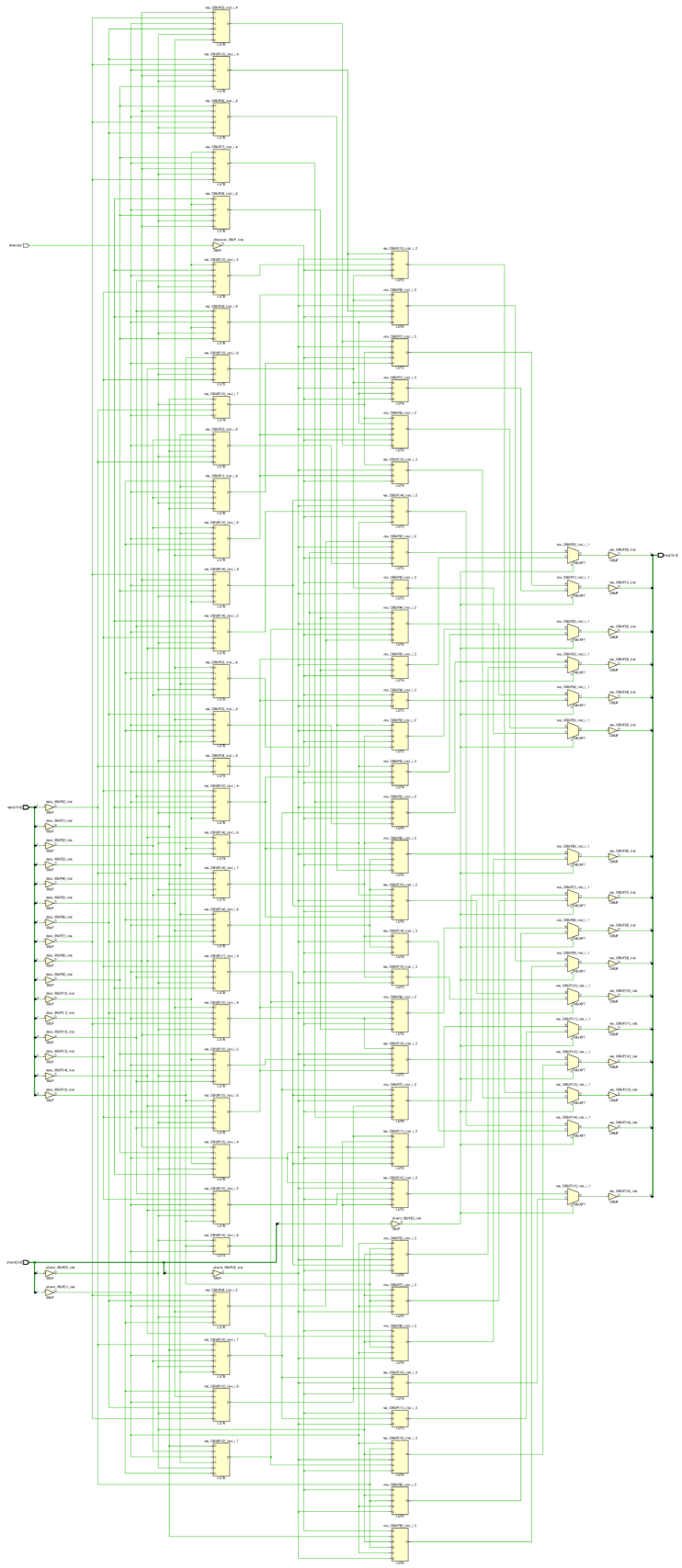
4.2.2. Multiplexor de 16 canales



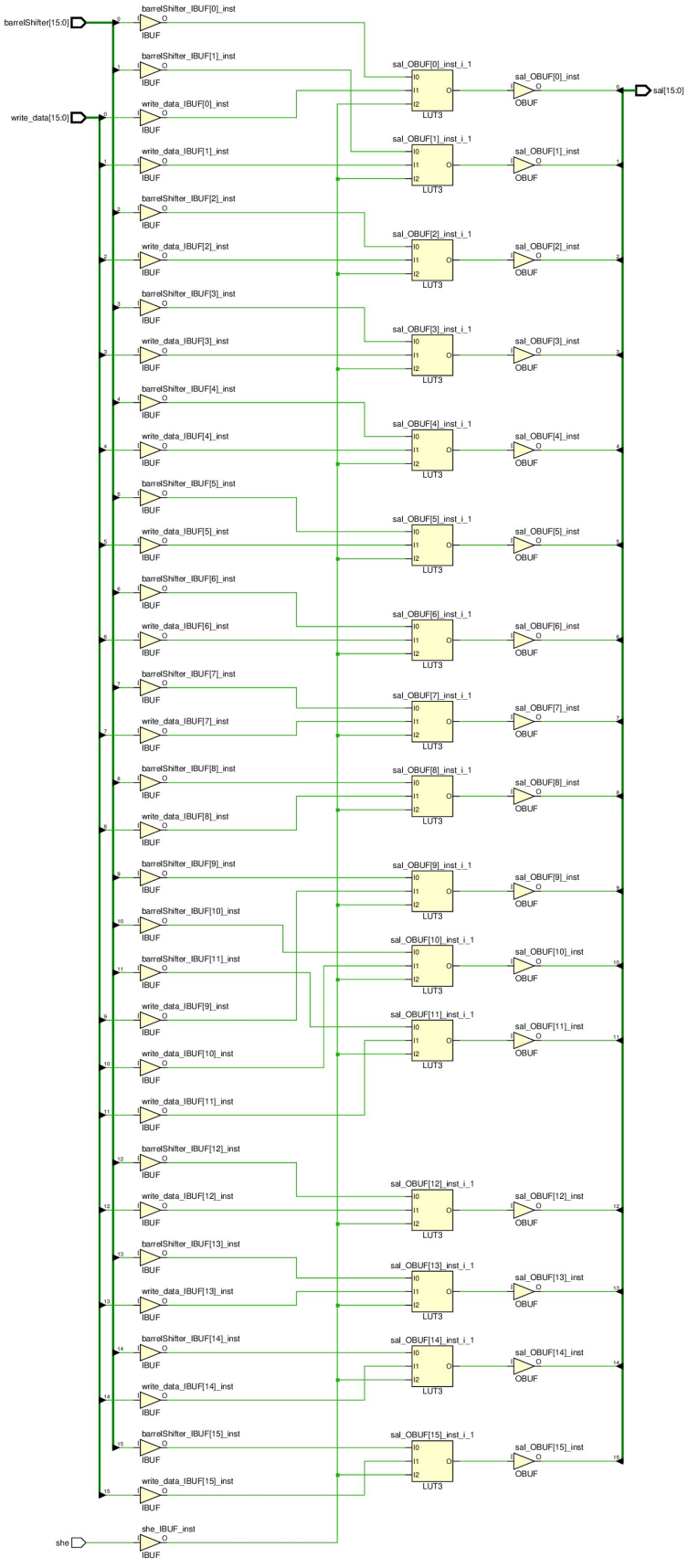
4.2.3. Demultiplexor de 16 canales de salida



4.2.4. Barrel-Shifter



4.2.5. Multiplexor de 2 canales



4.2.6. Archivo de Registros

Diagrama comprimido

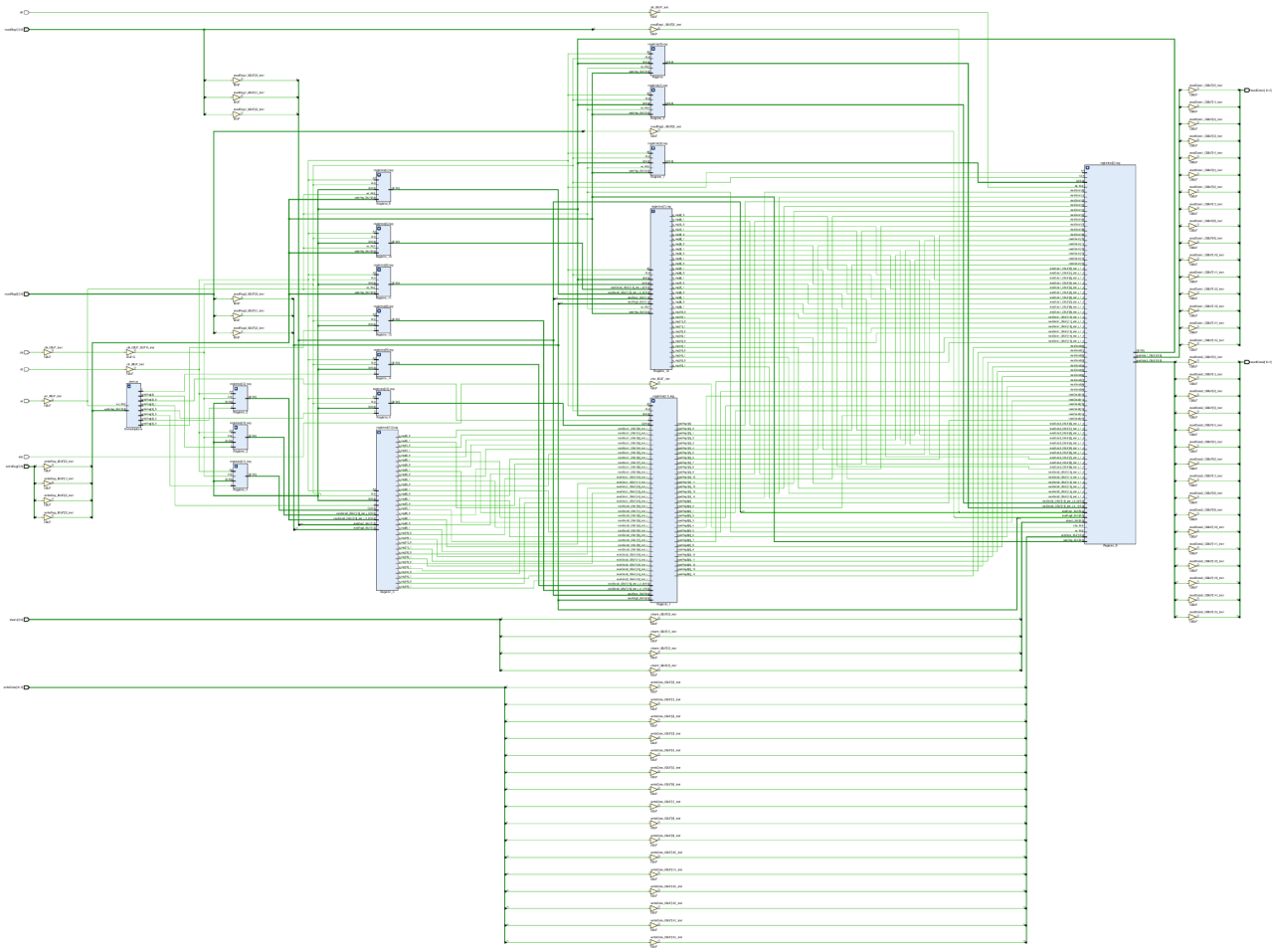


Diagrama expandido