Reporte de practica 6

González Pardo Adrian

Marzo 2020

1. Código VHDL

1.1. Mux de 16 canales

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

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

architecture Behavioral of muxGral is
begin
    process(selectMux)
        begin
            case selectMux is
                    when "0000"  =>outMux<= chanel_0;
                    when "0001"  =>outMux<= chanel_1;
                    when "0010"  =>outMux<= chanel_2;
                    when "0011"  =>outMux<= chanel_3;
                    when "0100"  =>outMux<= chanel_4;
                    when "0101"  =>outMux<= chanel_5;
                    when "0110"  =>outMux<= chanel_6;
                    when "0111"  =>outMux<= chanel_7;
                    when "1000"  =>outMux<= chanel_8;
                    when "1001"  =>outMux<= chanel_9;
                    when "1010"  =>outMux<= chanel_10;
                    when "1011"  =>outMux<= chanel_11;
                    when "1100"  =>outMux<= chanel_12;
                    when "1101"  =>outMux<= chanel_13;
                    when "1110"  =>outMux<= chanel_14;
                    when others => outMux<= chanel_15;
                end case;
    end process;
end Behavioral;
```

1.2. Mux de 2 canales

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux2 is
    Port(p,s: in STD_LOGIC_VECTOR (15 downto 0);
         sel : in STD_LOGIC;
         sout : out STD_LOGIC_VECTOR (15 downto 0));
end mux2;

architecture Behavioral of mux2 is
begin
    process(sel)
        begin
        if (sel = '0')then
            sout <= p;
        else
            sout <= s;
        end if;
    end process;
end Behavioral;
```

1.3. Demux de 16 canales

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity demuxG is
    Port(output:out std_logic_vector(15 downto 0);
        input:in  std_logic;
        selector:in std_logic_vector(3 downto 0));
end demuxG;

architecture Behavioral of demuxG is

begin
    process(selector,input)
        begin
            if(input = '1')then
                case selector is
                    when "0000"  =>output<= "0000000000000001";
                    when "0001"  =>output<= "0000000000000010";
                    when "0010"  =>output<= "0000000000000100";
                    when "0011"  =>output<= "0000000000001000";
                    when "0100"  =>output<= "0000000000010000";
                    when "0101"  =>output<= "0000000000100000";
                    when "0110"  =>output<= "0000000001000000";
                    when "0111"  =>output<= "0000000010000000";
                    when "1000"  =>output<= "0000000100000000";
                    when "1001"  =>output<= "0000001000000000";
                    when "1010"  =>output<= "0000010000000000";
                    when "1011"  =>output<= "0000100000000000";
                    when "1100"  =>output<= "0001000000000000";
                    when "1101"  =>output<= "0010000000000000";
                    when "1110"  =>output<= "0100000000000000";
                    when others => output<= "1000000000000000";
                end case;
            else
                output<="0000000000000000";
            end if;
    end process;
end Behavioral;
```

1.4. Registro de 16 bits

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

```

```
entity 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 registro;

architecture Behavioral of registro is
begin
     process(clr,clk,l,d)
    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;
```

1.5. Archivo de registros

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


entity archRegistros is
    Port(writeData : in STD_LOGIC_VECTOR (15 downto 0);
          writeReg : in STD_LOGIC_VECTOR (3 downto 0);
          readReg1 : in STD_LOGIC_VECTOR (3 downto 0);
          readReg2 : in STD_LOGIC_VECTOR (3 downto 0);
          shamt : in STD_LOGIC_VECTOR (3 downto 0);
          WR,CLK : in STD_LOGIC;
          SHE,DIR,clr : in STD_LOGIC;
          readData1 : inout STD_LOGIC_VECTOR (15 downto 0);
          readData2 : out STD_LOGIC_VECTOR (15 downto 0));
end archRegistros;

architecture Behavioral of archRegistros is
    component registro
    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;

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

    component muxGral
    Port (chanel_0:in std_logic_vector(15 downto 0);
          chanel_1:in std_logic_vector(15 downto 0);
          chanel_2:in std_logic_vector(15 downto 0);
          chanel_3:in std_logic_vector(15 downto 0);
          chanel_4:in std_logic_vector(15 downto 0);
          chanel_5:in std_logic_vector(15 downto 0);
          chanel_6:in std_logic_vector(15 downto 0);
          chanel_7:in std_logic_vector(15 downto 0);
          chanel_8:in std_logic_vector(15 downto 0);
          chanel_9:in std_logic_vector(15 downto 0);
          chanel_10:in std_logic_vector(15 downto 0);
          chanel_11:in std_logic_vector(15 downto 0);
          chanel_12:in std_logic_vector(15 downto 0);
          chanel_13:in std_logic_vector(15 downto 0);
```

```
          chanel_14:in std_logic_vector(15 downto 0);
          chanel_15:in std_logic_vector(15 downto 0);
          selectMux: in STD_LOGIC_VECTOR (3 downto 0);
          outMux :out std_logic_vector(15 downto 0) );
    end component;

    component demuxG
    Port(output:out std_logic_vector(15 downto 0);
        input:in  std_logic;
        selector:in std_logic_vector(3 downto 0));
    end component;

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

    signal BS_out :std_logic_vector(15 downto 0);--salida del barrel shifter
    signal WRSHE_out :std_logic_vector(15 downto 0);
    signal WR_writeREG:std_logic_vector(15 downto 0);
    TYPE banco is array (0 to 15) of std_logic_vector(15 downto 0);
    signal q:banco;

begin
    WR_writeregister: demuxG Port map(
        output => WR_writeREG,
        input => WR,
        selector => writeReg
    );

    Barrel_shifter:barrel Port map(
        dato => readData1,
        res => BS_out,
        dir => DIR,
        s => shamt
    );

    entrada_registros: mux2 port map(
        sel => SHE,
        p => writeData,
        s => BS_out,
        sout =>WRSHE_out
    );

    ciclo: for i in 0 to 15 generate
        Registros:registro PORT MAP(
        d=>WRSHE_out,
        q => q(i),
        clr =>clr,
        clk => CLK,
        L =>WR_writeREG(i)
        );
    end generate;

    muxreadData1:muxGral Port map(
        chanel_0 => q(0),
        chanel_1 => q(1),
        chanel_2 => q(2),
        chanel_3 => q(3),
        chanel_4 => q(4),
        chanel_5 => q(5),
        chanel_6 => q(6),
        chanel_7 => q(7),
        chanel_8 => q(8),
        chanel_9 => q(9),
        chanel_10 => q(10),
```

```
        chanel_11 => q(11),
        chanel_12 => q(12),
        chanel_13 => q(13),
        chanel_14 => q(14),
        chanel_15 => q(15),
        selectMux => readReg1,
        outMux =>readData1
    );
    muxreadData2:muxGral Port map(
        chanel_0 => q(0),
        chanel_1 => q(1),
        chanel_2 => q(2),
        chanel_3 => q(3),
        chanel_4 => q(4),
        chanel_5 => q(5),
        chanel_6 => q(6),
        chanel_7 => q(7),
        chanel_8 => q(8),
        chanel_9 => q(9),
        chanel_10 => q(10),
        chanel_11 => q(11),
        chanel_12 => q(12),
        chanel_13 => q(13),
        chanel_14 => q(14),
        chanel_15 => q(15),
        selectMux => readReg2,
        outMux =>readData2
    );
end Behavioral;
```

2. Test-Bench VHDL Código

2.1. Test-Bench registro

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

entity registroTB is
--  Port ( );
end registroTB;

architecture Behavioral of registroTB is
component registro
    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;
constant periodo : time := 10 ns;
begin
    tb:registro port map(
        d => d,
        q => q,
        clr => clr,
        clk => clk,
        l => l
    );
    reloj : process
        begin
            clk <= '0';
            wait for 5 ns;
            clk <= '1';
            wait for 5 ns;
    end process;
```

```
    simulacion:process
        begin
            clr <= '1';
            l <= '0';
            d <= "0001110011110000";
            wait for 40 ns;
            clr <= '0';
            l <= '1';
            d <= "0001110011110000";
            wait for 40 ns;
            clr <= '0';
            l <= '0';
            d <= "0000010011110000";
            wait for 40 ns;
            clr <= '0';
            l <= '1';
            d <= "1111111111111111";
            wait for 40 ns;
            clr <= '1';
            l <= '0';
            d <= "0001110011110000";
            wait for 40 ns;
            wait;
    end process;
end Behavioral;
```

2.2. Test-Bench barrel shifter

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

entity barrelTB is
--  Port ( );
end barrelTB;

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

signal dato : STD_LOGIC_VECTOR (15 downto 0);
signal res :  STD_LOGIC_VECTOR (15 downto 0);
signal dir: std_logic;
signal s :  STD_LOGIC_VECTOR (3 downto 0);
begin
    u1 : Barrel
    Port map (
        dato => dato,
        res  => res,
        dir =>dir,
        s => s
        );

    simulacion : process
        begin
        dato <= "0000000010010111";
        s <= "0011";
        dir <='0';
        wait for 10 ns;
        dato <= "0000110010010111";
        s <= "1011";
        dir <='0';
        wait for 10 ns;
        dato <= "1111111110011111";
```

```
        s <= "0011";
        dir <='1';
        wait for 10 ns;
        dato <= "1001011110011111";
        s <= "1011";
        dir <='1';
        wait for 10 ns;
        wait;
    end process;



end Behavioral;
```

2.3. Test-Bench archivo de registros

```
LIBRARY ieee;
LIBRARY STD;
USE STD.TEXTIO.ALL;
USE ieee.std_logic_TEXTIO.ALL;   --PERMITE USAR STD_LOGIC
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_UNSIGNED.ALL;
USE ieee.std_logic_ARITH.ALL;


entity archRegTB is
--  Port ( );
end archRegTB;

architecture Behavioral of archRegTB is
    component archRegistros
    Port(writeData : in STD_LOGIC_VECTOR (15 downto 0);
           writeReg : in STD_LOGIC_VECTOR (3 downto 0);
           readReg1 : in STD_LOGIC_VECTOR (3 downto 0);
           readReg2 : in STD_LOGIC_VECTOR (3 downto 0);
           shamt : in STD_LOGIC_VECTOR (3 downto 0);
           WR,CLK : in STD_LOGIC;
           SHE,DIR,clr : in STD_LOGIC;
           readData1 : inout STD_LOGIC_VECTOR (15 downto 0);
           readData2 : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    --inputs
    signal writeData :  STD_LOGIC_VECTOR (15 downto 0);
    signal writeReg :  STD_LOGIC_VECTOR (3 downto 0);
    signal readReg1 :  STD_LOGIC_VECTOR (3 downto 0);
    signal readReg2 :  STD_LOGIC_VECTOR (3 downto 0);
    signal shamt :  STD_LOGIC_VECTOR (3 downto 0);
    signal WR,CLK :  STD_LOGIC := '0';
    signal SHE,DIR,clr :  STD_LOGIC;

    --outputs
    signal readData1 :  STD_LOGIC_VECTOR (15 downto 0);
    signal readData2 :  STD_LOGIC_VECTOR (15 downto 0);

    --clock
    constant CLK_period : time := 10 ns;

begin
    tb_AR:archRegistros port map(
        writeData =>writeData,
        writeReg => writeReg,
        readReg1 => readReg1,
        readReg2 => readReg2,
        shamt => shamt,
        WR => WR,
        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/practicasVivado/registros/input.
        txt", READ_MODE);
85          file_open(ARCH_SAL, "/media/d3vcr4ck/externData/materias-Sem20_2/
        arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/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
```

```
        Hread(LINEA_ENT,VAR_WD);
        writeData <= VAR_WD;
        Hread(LINEA_ENT,VAR_WrRe);
        writeReg <= VAR_WrRe;
        Hread(LINEA_ENT,VAR_RR1);
        readReg1 <= VAR_RR1;
        Hread(LINEA_ENT,VAR_RR2);
        readReg2 <= VAR_RR2;
        Hread(LINEA_ENT,VAR_shamt);
        shamt <= VAR_shamt;
        read(LINEA_ENT,VAR_WR);
        WR <= VAR_WR;
        read(LINEA_ENT,VAR_SHE);
        SHE <= VAR_SHE;
        read(LINEA_ENT,VAR_DIR);
        DIR <= VAR_DIR;
        read(LINEA_ENT,VAR_clr);
        clr <= VAR_clr;

       --wait until rising_edge(CLK);
        wait for 10ns;
        VAR_RD1 := readData1;
        VAR_RD2 := readData2;


        HWRITE(LINEA_SAL,VAR_RR1, right, 5);
        HWRITE(LINEA_SAL,VAR_RR2, right, 5);
        WRITE(LINEA_SAL,VAR_shamt, right, 5);
        HWRITE(LINEA_SAL,VAR_WrRe, right, 5);
        HWRITE(LINEA_SAL,VAR_WD, right, 5);
      WRITE(LINEA_SAL,VAR_WR,right,5);
        WRITE(LINEA_SAL,VAR_SHE, right, 5);
        WRITE(LINEA_SAL,VAR_DIR, right, 5);
        HWRITE(LINEA_SAL,VAR_RD1, right, 5);
        HWRITE(LINEA_SAL,VAR_RD2, right, 5);

        writeline(ARCH_SAL,LINEA_SAL);
      end loop;
      file_close(ARCH_ENT);
      file_close(ARCH_SAL);
      wait;
    end process;
end Behavioral;
```

2.3.1. Archivos de entrada y salida para esta sección

Archivo de entrada (input.txt)

```
0f86 5 8 4 0 1 0 1 1
0089 1 1 0 0 1 0 0 0
0072 2 2 1 0 1 0 0 0
0123 3 3 0 0 1 0 0 0
0053 4 4 0 0 1 0 0 0
ffff f 1 2 0 0 0 0 0
eeee e 3 4 0 0 0 0 0
1234 2 1 2 3 1 1 0 0
5555 4 3 4 5 1 1 1 0
ffff f 1 2 0 0 0 0 0
eeee e 3 4 0 0 0 0 0
0f86 5 8 4 0 1 0 1 1



WriteData|writeReg|RR1|RR2|shamt|WR|SHE|DIR|clr
```

Archivo de salida (output.txt)

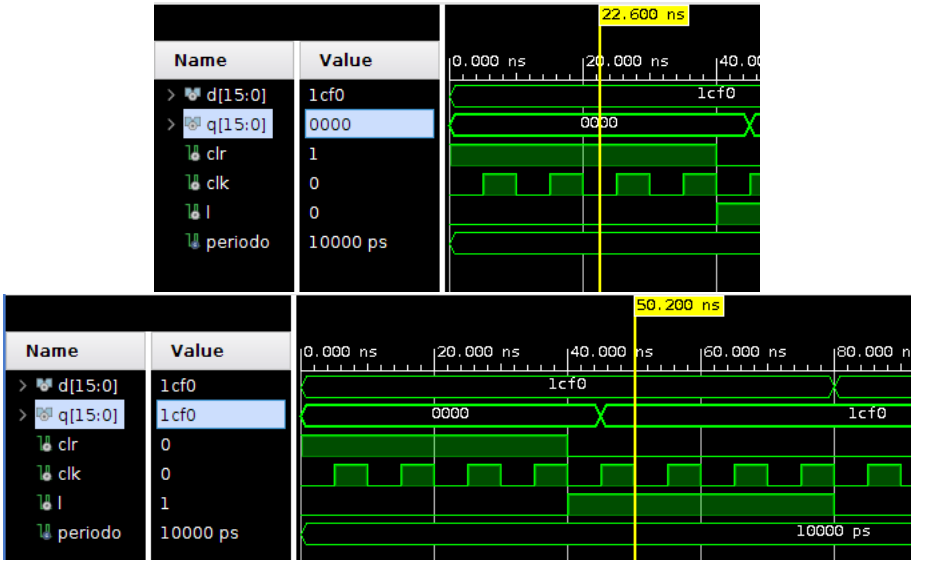
```
_RR1__    _RR2  SHAMT   _WREG   _WD__   _WR__   _SHE_   _DIR_   _RD1_   _RD2_
   8     4 0000     5 0F86     1     0     1 0000 0000
   1     0 0000     1 0089     1     0     0 0089 0000
   2     1 0000     2 0072     1     0     0 0072 0089
   3     0 0000     3 0123     1     0     0 0123 0000
   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
```

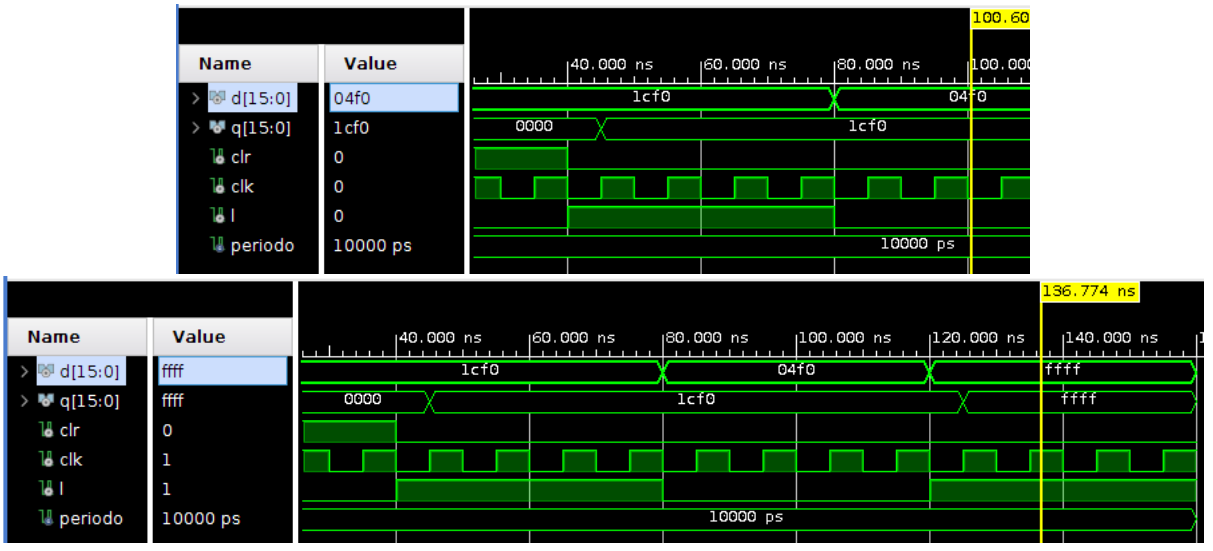
2.3.2. Tabla de resultados de la salida

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

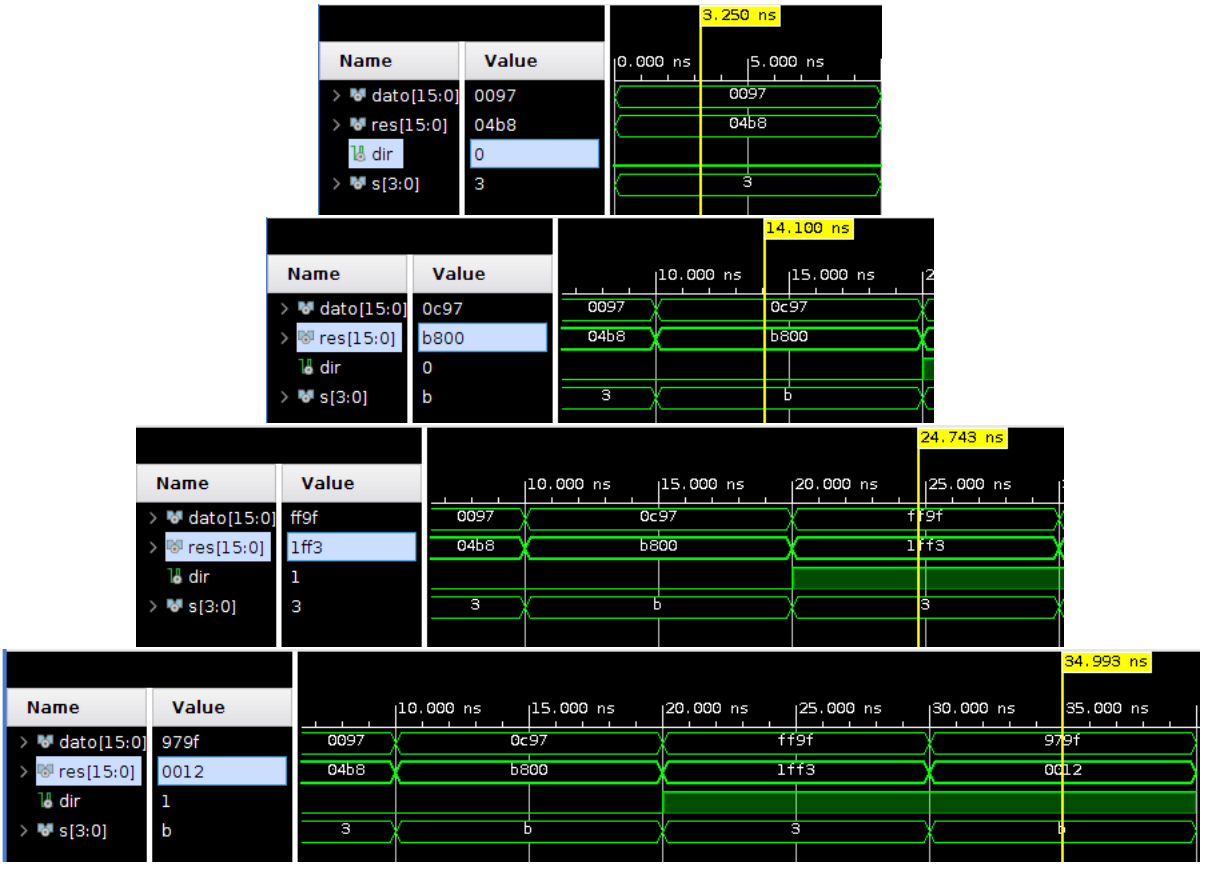
3. Simulaciones

3.1. Registro



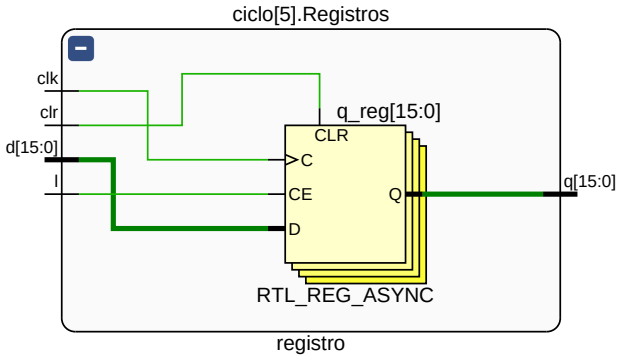


3.2. Barrel-Shifter

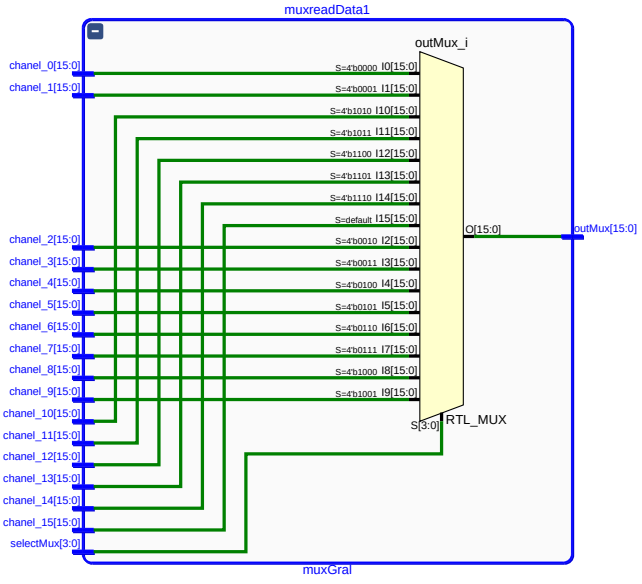


4. Diagramas RTL

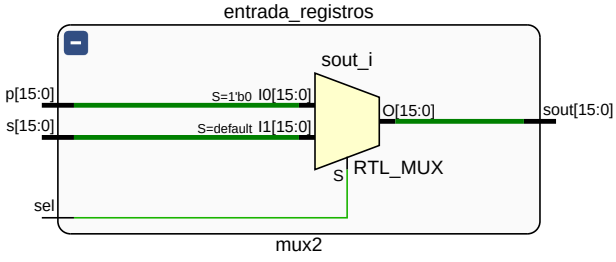
4.1. Registro



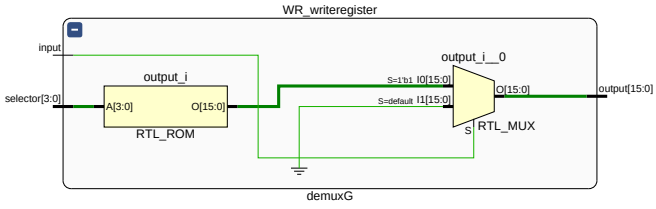
4.2. Mux 16 canales



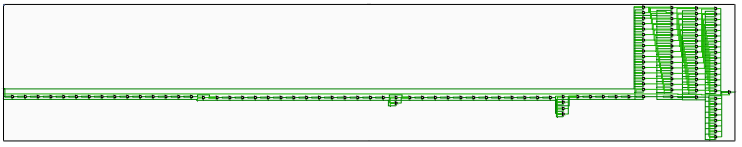
4.3. Mux 2 canales



4.4. Demux


WR_writeregister
input
selector[3:0]
output_i
A[3:0]
O[15:0]
RTL_ROM
output_i__0
S=1'b1 I0[15:0]
S=default I1[15:0]
O[15:0]
S
RTL_MUX
output[15:0]
demuxG

4.5. Barrel-Shifter



4.6. Archivo de Registros