Reporte de practica 2

González Pardo Adrian

Febrero 2020

1. Código VHDL

```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5
6  entity sumador is
7      Port ( a,b,cin : in STD_LOGIC;
8             s : out STD_LOGIC;
9             cout : out STD_LOGIC);
10 end sumador;
11
12 architecture Behavioral of sumador is
13 begin
14     s<=a xor b xor cin;
15     cout<= (a and cin)or(b and cin)or(a and b);
16
17 end Behavioral;
```

Código 1: Descripción del modulo sumador de 1 solo bit

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity sumadorNBits is
5  generic(
6      n: integer:= 8);
7      Port ( a,b : in STD_LOGIC_VECTOR (n-1 downto 0);
8             cin : in STD_LOGIC;
9             s : out STD_LOGIC_VECTOR (n-1 downto 0);
10            cout : out STD_LOGIC);
11 end sumadorNBits;
12
13 architecture Behavioral of sumadorNBits is
14 component sumador is
15     Port ( a,b,cin : in STD_LOGIC;
16            s : out STD_LOGIC;
17            cout : out STD_LOGIC);
```

```
end component;
signal c:std_logic_vector(n downto 0);
signal eb:std_logic_vector(n-1 downto 0);
begin
c(0)<=cin;
    ciclo: for i in 0 to n-1 generate
        eb(i)<=b(i) xor c(0);
        bit1: sumador Port map(
            a=> a(i),
            b=>eb(i),
            cin=>c(i),
            s=>s(i),
            cout=>c(i+1)
        );
    end generate;
    cout<=c(n);


end Behavioral;
```

Código 2: A través de uso de componentes de código para la descripción del sumador/restador en cascada de 8 bits

2. Test-Bench VHDL Código

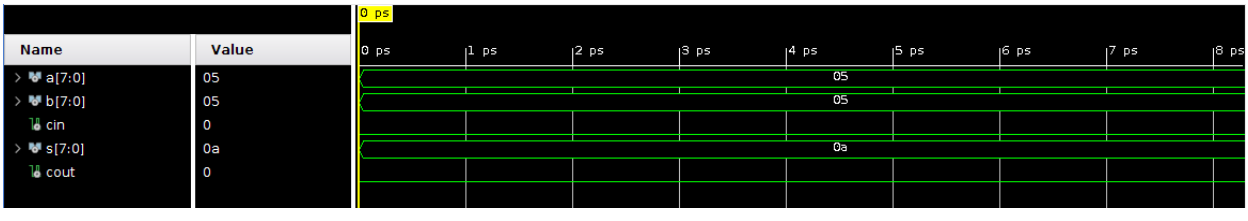
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity simuSumador is
--  Port ( );
end simuSumador;

architecture Behavioral of simuSumador is
component sumadorNBits is
    Port ( a,b : in STD_LOGIC_VECTOR (7 downto 0);
           cin : in STD_LOGIC;
           s : out STD_LOGIC_VECTOR (7 downto 0);
           cout : out STD_LOGIC);
end component;
signal a :STD_LOGIC_VECTOR (7 downto 0) := x"00";
signal b :STD_LOGIC_VECTOR (7 downto 0) := x"00";
signal cin : STD_LOGIC := '0';
signal s : STD_LOGIC_VECTOR (7 downto 0);
signal cout : STD_LOGIC;
begin
sumado : sumadorNBits
    Port map (
    a => a,
    b => b,
    cin => cin,
    s => s,
    cout => cout
    );
```

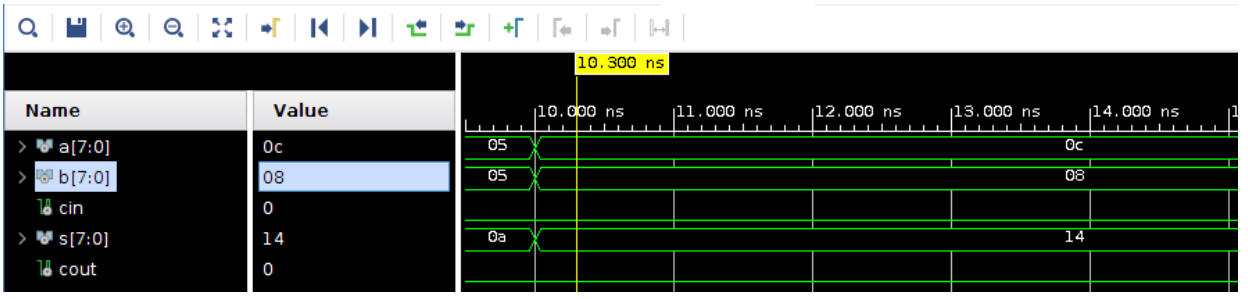
```
p1 : process
begin
    cin <= '0';
    a <= x"05";
    b <= x"05";
    wait for 10 ns;
    a <= x"0C";
    b <= x"08";
    wait for 10 ns;
    a <= x"09";
    b <= x"05";
    wait for 10 ns;
    cin <= '1';
    a <= x"0A";
    b <= x"09";
    wait for 10 ns;
    cin <= '0';
    a <= x"04";
    b <= x"02";
    wait for 10 ns;
    cin <= '1';
    a <= x"07";
    b <= x"09";
    wait for 10 ns;
    a <= x"0F";
    b <= x"0F";
    wait for 10 ns;
    a <= x"0B";
    b <= x"08";
    wait for 10 ns;
    a <= x"0A";
    b <= x"09";
    wait for 10 ns;
    a <= x"01";
    b <= x"04";
    wait;
end process;
end Behavioral;
```

Código de simulación

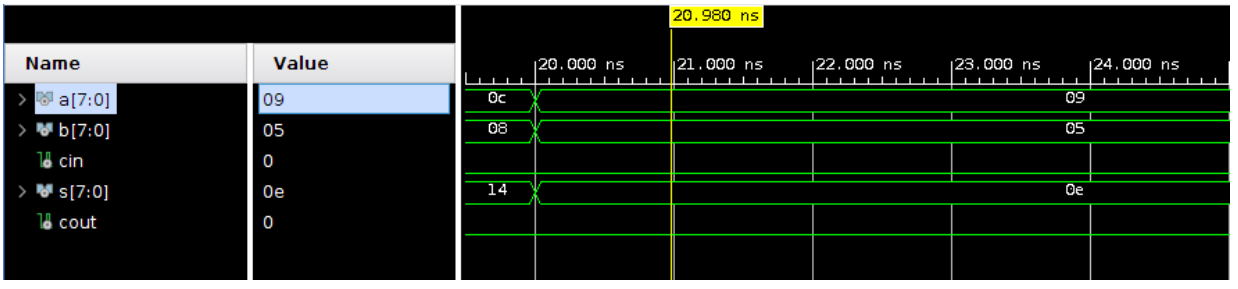
Anexo de fotos de la simulación de impulsos



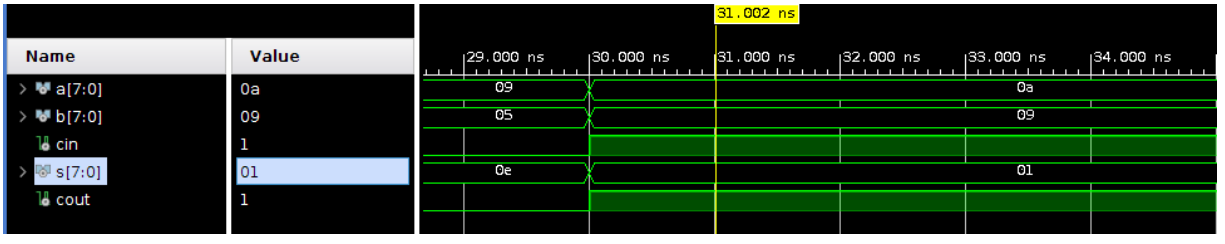
Primer parte con valores hexadecimales equivales a valores decimales: $a = 5_{10}$ & $b = 5_{10}$ *con salida* $s = A_{16} = 10_{10}$



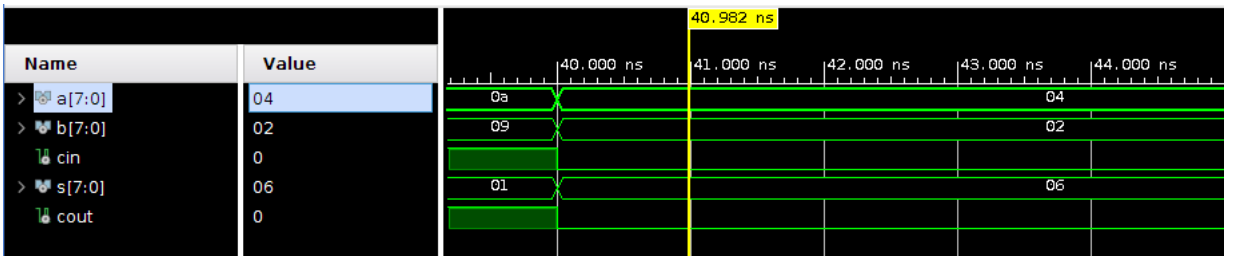
Segunda parte con valores hexadecimales equivales a valores decimales: $a = C_{16} = 12_{10}$ & $b = 8_{10}$ *con salida* $s = 14_{16} = 20_{10}$



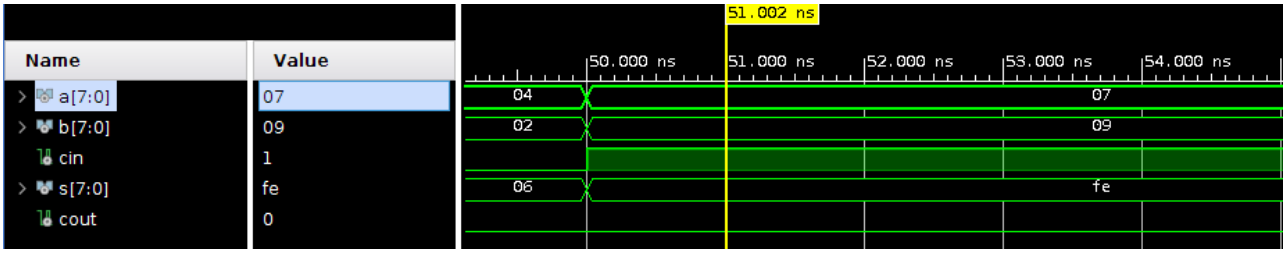
Tercer parte con valores hexadecimales equivales a valores decimales: $a = 9_{10}$ & $b = 5_{10}$ *con salida* $s = E_{16} = 14_{10}$



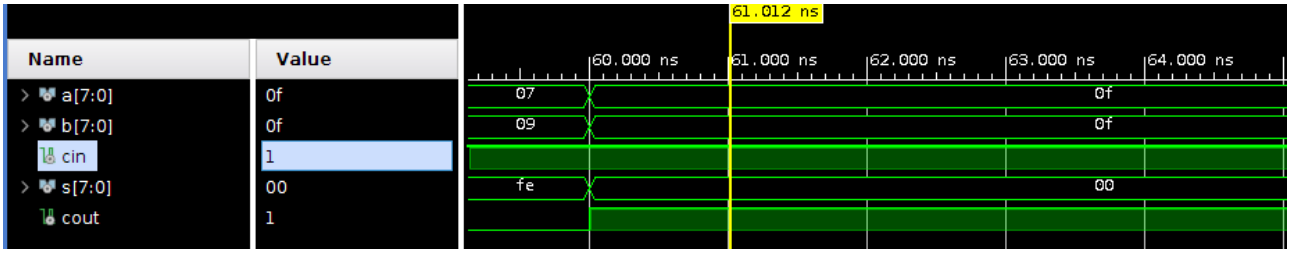
Cuarta parte con valores hexadecimales equivales a valores decimales: $a = A_{16} = 10_{10}$, $b = 9_{10}$ & $cin = 1_{10}$ *con salida* $s = 1_{10}$ *y un cout* $= 1_{10}$



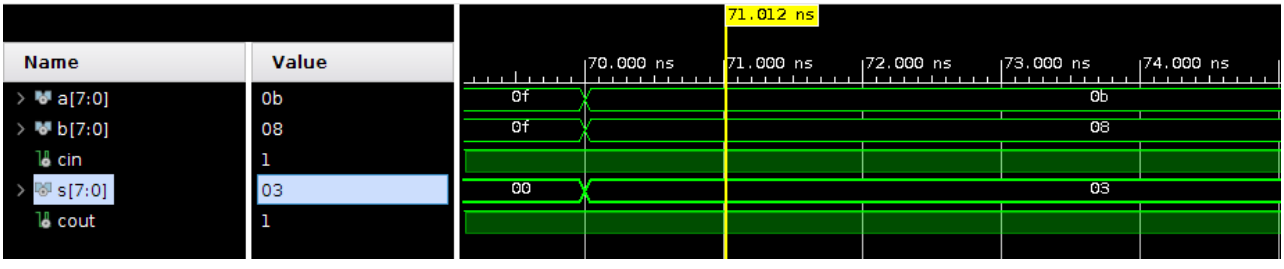
Quinta parte con valores hexadecimales equivales a valores decimales: $a = 4_{10}$, $b = 2_{10}$ & $cin = 0_{10}$ *con salida* $s = 6_{10}$



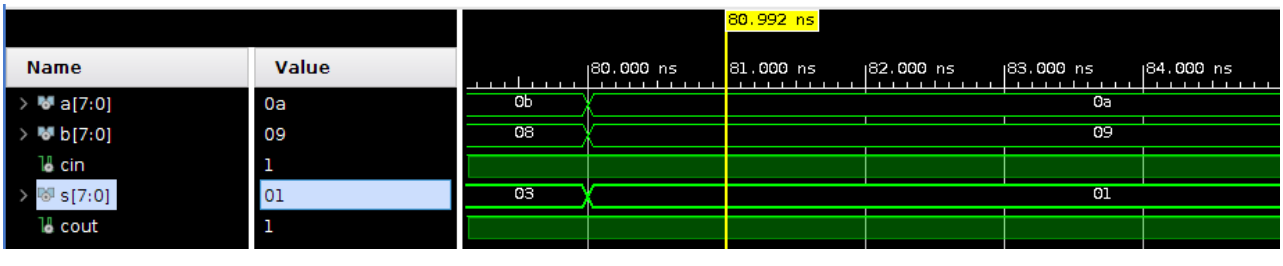
Sexta parte con valores hexadecimales equivales a valores decimales: $a = 7_{10}$, $b = 9_{10}$ & $cin = 1_{10}$ *con salida* $s = FE_{16} = 254_{10\_C2} = -2_{10}$ *y un cout* $= 0_{10}$



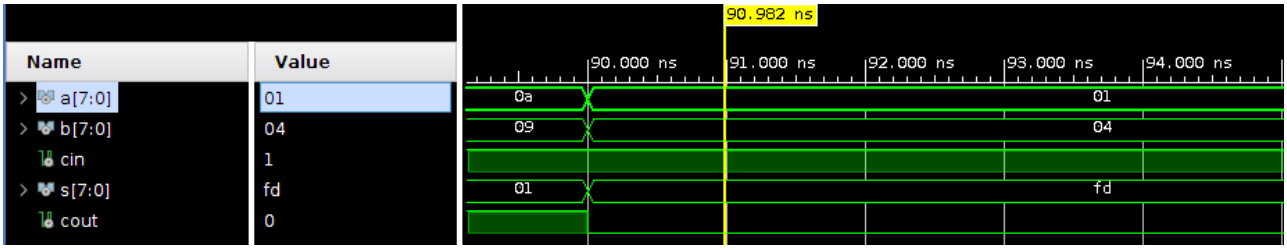
Septima parte con valores hexadecimales equivales a valores decimales: $a = F_{16} = 15_{10}$, $b = F_{16} = 15_{10}$ & $cin = 1_{10}$ *con salida* $s = 0_{10}$ *y un cout* $= 1_{10}$



Octava parte con valores hexadecimales equivales a valores decimales: $a = B_{16} = 11_{10}$, $b = 8_{10}$ & $cin = 1_{10}$ *con salida* $s = 3_{10}$ *y un cout* $= 1_{10}$



Novena parte con valores hexadecimales equivales a valores decimales: $a = A_{16} = 10_{10}$, $b = 9_{10}$ & $cin = 1_{10}$ *con salida* $s = 1_{10}$ *y un cout* $= 1_{10}$



Decima parte con valores hexadecimales equivales a valores decimales: $a = 1_{10}$, $b = 4_{10}$ & $cin = 1_{10}$ *con salida* $s = FD_{16} = 253_{10\_C2} = -3_{10}$ *y un cout* $= 0_{10}$

3. Tabla de resultados

Operación	A	B	S	Cout
Suma	5	5	10	0
Suma	12	8	20	0
Suma	9	5	14	0
Resta	10	9	1	1
Suma	4	2	6	0
Resta	7	9	254_C2 = -2	0
Resta	15	15	0	1
Resta	11	8	3	1
Resta	10	9	1	1
Resta	1	4	253_C2 = -3	0

4. Diagrama RTL

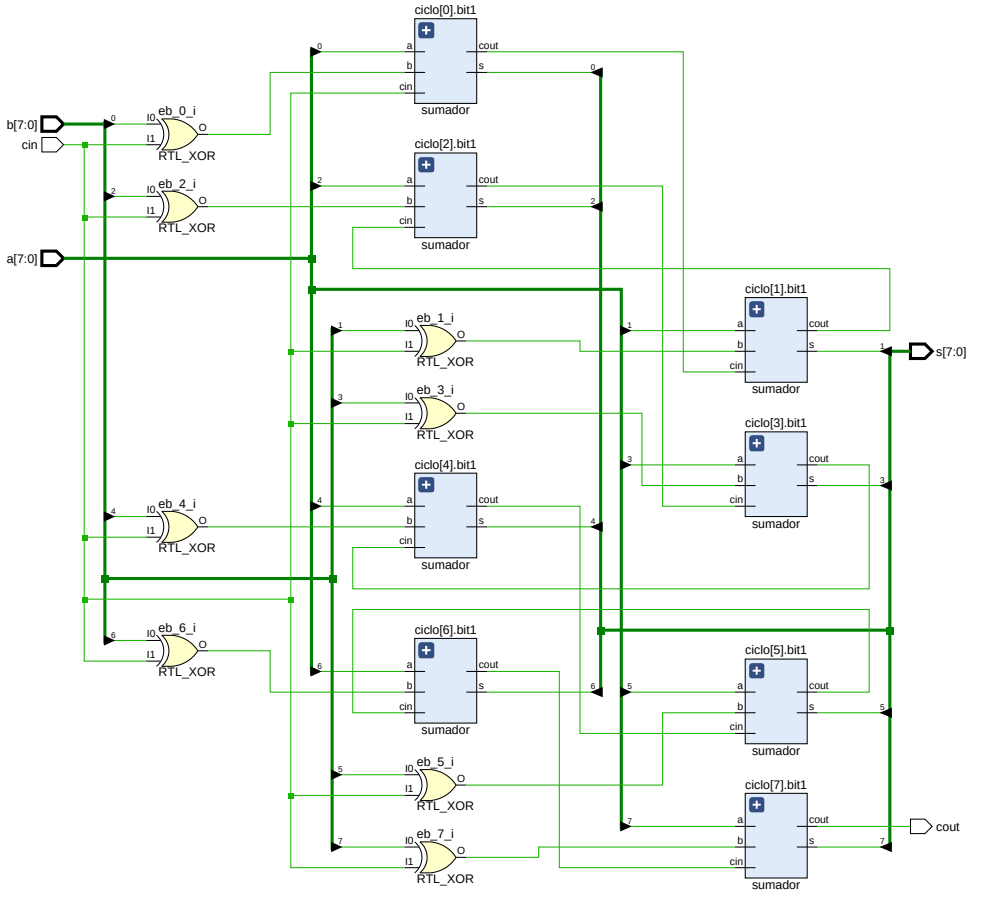


Diagrama RTL del archivo VHDL del sumador/restador en cascada de 8 bits

5. Diagrama de Síntesis

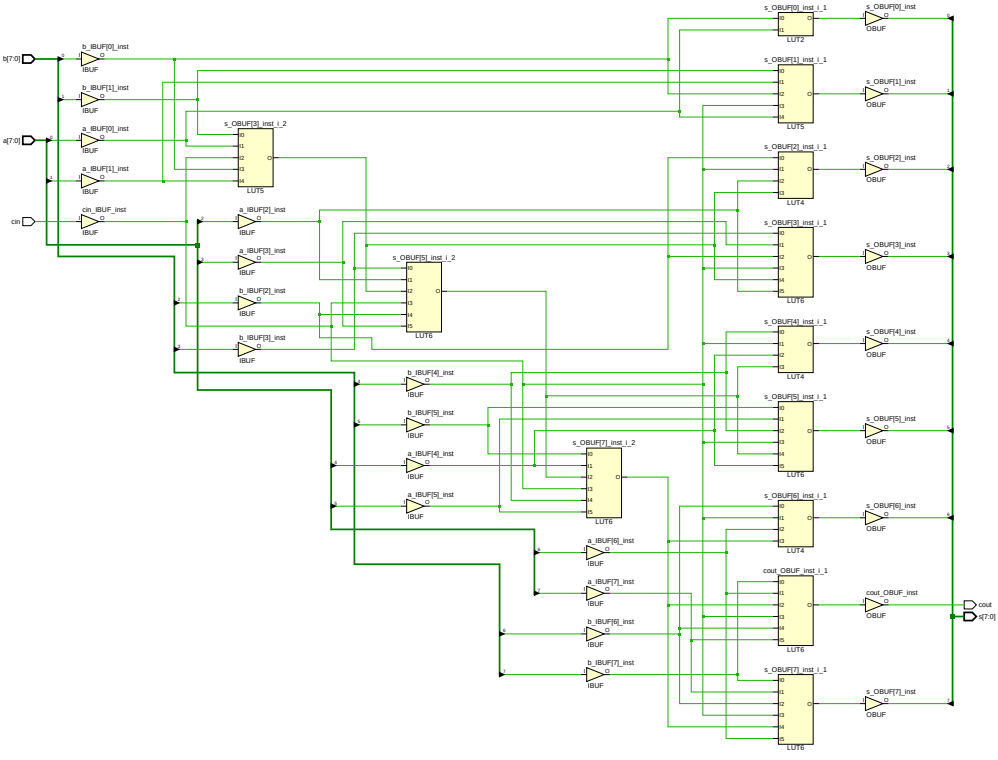


Diagrama de Sintesis del archivo VHDL del sumador/restador en cascada de 8 bits