## Reporte de practica 3

#### González Pardo Adrian

Febrero 2020

## 1. Código VHDL

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
  entity practica is
      Port ( a,b : in STD_LOGIC_VECTOR (7 downto 0);
              cin : in STD_LOGIC;
              cout : out STD_LOGIC;
              s : out STD_LOGIC_VECTOR (7 downto 0));
  end practica;
9
  architecture Behavioral of practica is
  begin
      process(a,b,cin)
13
      variable P,g:std_logic_vector(7 downto 0);
14
      variable c:std_logic_vector(8 downto 0);
      variable auxc, auxa, auxb, auxd:std_logic;
      begin
          c(0) := cin;
18
          for i in 0 to 7 loop
               P(i) := a(i) xor b(i);
               g(i) := a(i) and b(i);
               s(i) \leftarrow P(i) xor c(i);
               auxc:='1';
24
               for j in 0 to i loop
26
                   auxc:= auxc and P(j);
               end loop;
               auxa:= auxc and C(0);
30
               auxd:='0';
               for k in 0 to i-1 loop
32
                   auxb:= '1';
33
                   for m in k+1 to i loop
34
                        auxb:= auxb and p(m);
                   end loop;
36
                   auxd:= auxd or (auxb and g(k));
```

Código fuente de sumador con acarreo anticipado

# 2. Test-Bench VHDL Código

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
4 entity simPrac2 is
5 -- Port ();
6 end simPrac2;
 architecture Behavioral of simPrac2 is
  component practica is
      Port(a,b : in STD_LOGIC_VECTOR (7 downto 0);
              cin : in STD_LOGIC;
11
              cout : out STD_LOGIC;
              s : out STD_LOGIC_VECTOR (7 downto 0));
14 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';
18 signal s: STD_LOGIC_VECTOR (7 downto 0);
19 signal cout: STD_LOGIC;
20 begin
21 sumaAnt: practica
     Port map (
22
           a = a,
23
           b = > b,
           cin=>cin,
           s=>s,
26
           cout => cout
      );
29
30 p2: process
31 begin
   cin <= '0';
   a \le x "05";
   b \le x 05;
34
   wait for 10 ns;
35
   a \le x " 0 C ";
   b \le x 07;
   wait for 10 ns;
38
   a \le x "09";
39
   b \le x 05;
   wait for 10 ns;
```

```
a \le x "0E";
b \le x "09";
   wait for 10 ns;
   a \le x 04;
   b \le x 02;
46
   wait for 10 ns;
47
   a \le x 07;
  b \le x 07;
49
   wait for 10 ns;
50
   a \le x " OF";
51
   b \le x 05;
   wait for 10 ns;
53
a \le x "0B";
   b<=x"08";
   wait for 10 ns;
   a \le x 01;
   b \le x 04;
58
   wait;
60 end process;
62 end Behavioral;
```

Código de simulación Anexo de fotos de la simulación a impulsos

		1.020 ns
Name	Value	0.000 ns  2.000 ns  4.000 ns  6.000 ns  8.000 ns
> <b>W</b> a[7:0]	05	05
> <b>W</b> b[7:0]	05	05
<b>l</b> ₄ cin	0	
> <b>W</b> s[7:0]	0a	Oa
¹⊌ cout	0	

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

			11.96 <mark>0 ns</mark>			
Name	Value	10.000 ns	12. <b>000</b> ns	14.000 ns	16.000 ns	18. <b>000</b> ns
> <b>₩</b> a[7:0]	0c			Ос		)
> 😽 b[7:0]	07			07		
⅓ cin	0					
> <b>W</b> s[7:0]	13			13		·
¹⊌ cout	0					

Segunda parte con valores hexadecimales equivales a valores decimales:  $a = C_{16} = 12_{10} \& b = 7_{10}$  con salida  $s = 13_{16} = 19_{10}$ 

		22.020 ns
Name	Value	20.000 ns   22.000 ns   24.000 ns   26.000 ns   28.000 ns
> <b>W</b> a[7:0]	09	( 09
> 💆 b[7:0]	05	05
¹⊌ cin	0	
> <b>W</b> s[7:0]	0e	0e
⅓ cout	0	
7⊌ cin > <b>₩</b> s[7:0]	0 0e	

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

			32.040 ns			
Name	Value	30.000 ns	32.000 ns	34.000 ns	36.000 ns	38. <b>000</b> ns
> <b>W</b> a[7:0]	0e			0e		)
> <b>W</b> b[7:0]	09	(		09		·
⅓ cin	0					
> <b>W</b> s[7:0]	17			17		·
l⊌ cout	0					

Cuarta parte con valores hexadecimales equivales a valores decimales:  $a=E_{16}=14_{10}~\&~b=9_{10}~con~salida~s=17_{16}=23_{10}$ 

			42.040 ns			
Name	Value	40.000 ns	42.000 ns	44.000 ns	46.000 ns	48. <b>000</b> ns
> <b>W</b> a[7:0]	04			04		
> ₩ b[7:0]	02			02		<b>'</b>
⅓ cin	0					
> <b>W</b> s[7:0]	06			06		
¹⊌ cout	0					

Quinta parte con valores hexadecimales equivales a valores decimales:  $a=4_{10}~\&~b=2_{10}$  con salida  $s=6_{10}$ 

			52.020 ns			
Name	Value	50.000 ns	52.000 ns	54.000 ns	56. <b>000</b> ns	58. <b>000</b> ns
> <b>W</b> a[7:0]	07			07		
> <b>W</b> b[7:0]	07			07		
l cin	0					
> <b>W</b> s[7:0]	0e			0e		
¹⊌ cout	0					

Sexta parte con valores hexadecimales equivales a valores decimales:  $a=7_{10}~\&~b=7_{10}~con$  salida  $s=E_{16}=14_{10}$ 

			62.020 ns			
Name	Value	60.000 ns	62. <b>000</b> ns	64.000 ns	66. <b>000</b> ns	68. <b>000</b> ns
> <b>₩</b> a[7:0]	Of			0f		
> ₩ b[7:0]	05			05		
⅓ cin	0					
> <b>⊌</b> s[7:0]	14			14		
¹⊌ cout	0					

Septima parte con valores hexadecimales equivales a valores decimales:  $a=F_{16}=15_{10}$  &  $b=5_{10}$  con salida  $s=14_{16}=20_{10}$ 

		72,020 ns
Name	Value	70.000 ns 72.000 ns 74.000 ns 76.000 ns 78.000 ns
> <b>₩</b> a[7:0]	0b	Ob
> Ե [7:0]	08	08
⅓ cin	0	
> <b>W</b> s[7:0]	13	13
¹⊌ cout	0	

Octava parte con valores hexadecimales equivales a valores decimales:  $a=B_{16}=11_{10}~\&~b=8_{10}~con~salida~s=13_{16}=19_{10}$ 

			82. <mark>020</mark> ns			
Name	Value	80.000 ns	82. <b>000</b> ns	84.000 ns	86. <b>000</b> ns	88. <b>000</b> ns
> <b>W</b> a[7:0]	01		01			
> 😽 b[7:0]	04	(			04	
l∄ cin	0					
> <b>W</b> s[7:0]	05		05			
™ cout	0					

Novena parte con valores hexadecimales equivales a valores decimales:  $a=1_{10}~\&~b=4_{10}$  con salida  $s=5_{10}$ 

## 3. Tabla de resultados

Operación	A	В	S	Cout
Suma	5	5	10	0
Suma	12	7	19	0
Suma	9	5	14	0
Suma	14	9	23	0
Suma	4	2	6	0
Suma	7	7	14	0
Suma	15	5	20	0
Suma	11	8	19	0
Suma	1	4	5	0

# 4. Diagrama RTL

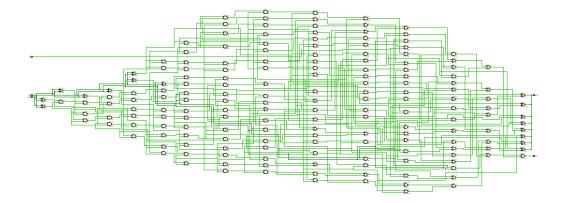


Diagrama RTL del archivo VHDL del sumador con acarreo anticipado de 8 bits

# 5. Diagrama de Síntesis

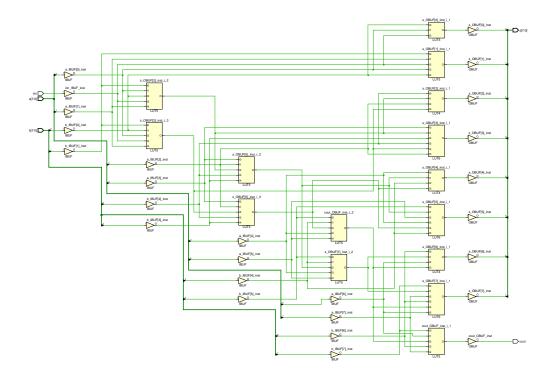


Diagrama de Síntesis del archivo VHDL del sumador con acarreo anticipado de 8 bits