Martínez Coronel Brayan Yosafat

# Código VHDL

## Sumador de 1 bit

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity sum\_1bit is

Port ( a, b, cin : in STD\_LOGIC;

s, cout : out STD\_LOGIC);

end sum\_1bit;

architecture Behavioral of sum\_1bit is

begin

s <= a xor b xor cin;

cout <= (a and b) or (a and cin) or (b and cin);

end Behavioral;

## ALU de 1 bit

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity alu\_1bit is

Port ( a, b, s\_a, s\_b, cin : in STD\_LOGIC;

alu\_op : in STD\_LOGIC\_VECTOR (1 downto 0);

r, cout : out STD\_LOGIC);

end alu\_1bit;

architecture Behavioral of alu\_1bit is

component sum\_1bit is

Port ( a, b, cin : in STD\_LOGIC;

s,cout : out STD\_LOGIC);

end component;

signal aux\_b, aux\_a, aux\_and, aux\_or, aux\_xor, aux\_sum : STD\_LOGIC;

begin

aux\_a <= (a and not s\_a) or (s\_a and not a);

aux\_b <= (b and not s\_b) or (s\_b and not b);

aux\_and <= aux\_a and aux\_b;

aux\_or <= aux\_a or aux\_b;

aux\_xor <= aux\_a xor aux\_b;

sum : sum\_1bit port map (

a => aux\_a,

b => aux\_b,

cin => cin,

s => aux\_sum,

cout => cout

);

process (aux\_and, aux\_or, aux\_xor, aux\_sum, alu\_op)

begin

case alu\_op is

when "00" => r <= aux\_and;

when "01" => r <= aux\_or;

when "10" => r <= aux\_xor;

when others => r <= aux\_sum;

end case;

end process;

end Behavioral;

## ALU de 4 bits

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity alu\_4bits is

Port ( A, B, op : in STD\_LOGIC\_VECTOR (3 downto 0);

OV, Z, C, N : out STD\_LOGIC;

R : inout STD\_LOGIC\_VECTOR (3 downto 0)

);

end alu\_4bits;

architecture Behavioral of alu\_4bits is

component alu\_1bit is

Port ( a, b, s\_a, s\_b, cin : in STD\_LOGIC;

alu\_op : in STD\_LOGIC\_VECTOR (1 downto 0);

r, cout : out STD\_LOGIC);

end component;

signal SEL : STD\_LOGIC\_VECTOR (3 downto 0);

signal carry : STD\_LOGIC\_VECTOR (4 downto 0);

begin

carry(0) <= op(2);

ciclo: for j in 0 to 3 generate

comp : alu\_1bit port map(

a => A(j),

b => B(j),

s\_a => op(3),

s\_b => op(2),

cin => carry(j),

alu\_op => op (1 downto 0),

r => R(j),

cout => carry(j+1)

);

end generate;

with op select

OV <= carry(3) xor carry(4) when "0011",

carry(3) xor carry(4) when "0111",

'0' when others;

with op select

C <= carry(4) when "0011",

carry(4) when "0111",

'0' when others;

N <= R(3);

Z <= not (R(0) or R(1) or R(2) or R(3));

end Behavioral;

# Código del TEST-BENCH

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity t\_alu\_4bits is

-- Port ( );

end t\_alu\_4bits;

architecture Behavioral of t\_alu\_4bits is

component alu\_4bits is

Port ( A, B, op : in STD\_LOGIC\_VECTOR (3 downto 0);

OV, Z, C, N : out STD\_LOGIC;

R : inout STD\_LOGIC\_VECTOR (3 downto 0)

);

end component;

signal A, B, op : STD\_LOGIC\_VECTOR (3 downto 0);

signal OV, Z, C, N : STD\_LOGIC;

signal R : STD\_LOGIC\_VECTOR (3 downto 0);

begin

test : alu\_4bits port map (

A => A,

B => B,

op => op,

OV => OV,

Z => Z,

C => C,

N => N,

R => R

);

process

begin

A <= "0101"; -- 5

B <= "1110"; -- -2

op <= "0011"; -- +

wait for 50 ns;

op <= "0111"; -- -

wait for 50 ns;

op <= "0000"; -- and

wait for 50 ns;

op <= "1101"; -- nand

wait for 50 ns;

op <= "0001"; -- or

wait for 50 ns;

op <= "1100"; -- nor

wait for 50 ns;

op <= "0010"; -- xor

wait for 50 ns;

op <= "1010"; --xnor

wait for 50 ns;

B <= "0111";

op <= "0011"; -- +

wait for 50 ns;

B <= "0101";

op <= "0111"; -- -

wait for 50 ns;

op <= "1101"; -- nand

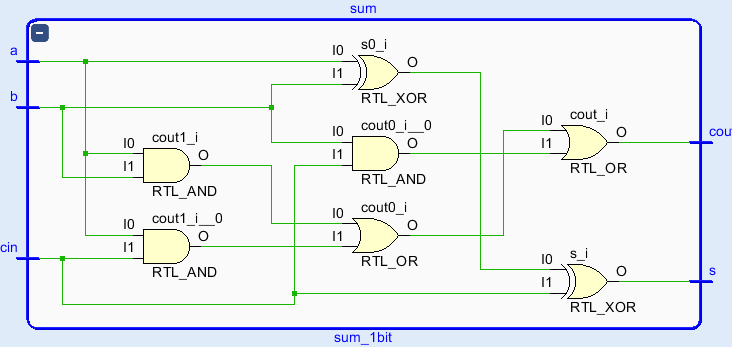
wait;

end process;

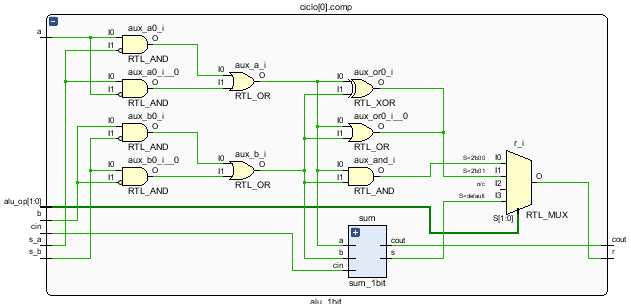
end Behavioral;

# Diagramas RTL

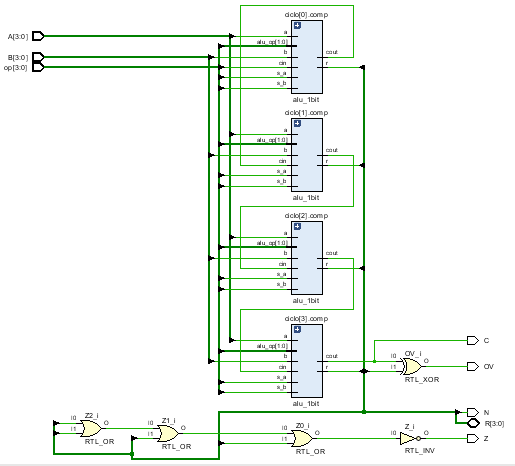
## Sumador de 1 bit



## ALU de 1 bit



## ALU 4 bits



# Simulación

