**MUX**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity MyMux8\_1 is

Port ( Din : in STD\_LOGIC\_VECTOR (7 downto 0);

Sel : in STD\_LOGIC\_VECTOR (2 downto 0);

Dout : out STD\_LOGIC;

Dout1 : out STD\_LOGIC);

end MyMux8\_1;

architecture Behavioral of MyMux8\_1 is

begin

Dout<=Din(0) when Sel="000" else

Din(1) when Sel="001" else

Din(2) when Sel="010" else

Din(3) when Sel="011" else

Din(4) when Sel="100" else

Din(5) when Sel="101" else

Din(6) when Sel="110" else

Din(7) when Sel="111" else 'X';

with Sel select

Dout1<=Din(0) when "000",

Din(1) when "001",

Din(2) when "010",

Din(3) when "011",

Din(4) when "100",

Din(5) when "101",

Din(6) when "110",

Din(7) when "111",

'X' when others;

end Behavioral;

**FA**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity FA is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

Cout : out STD\_LOGIC;

S : out STD\_LOGIC);

end FA;

architecture Behavioral of FA is

begin

cout<=(A and B) or ((A xor B) and Cin);

S<=(A xor B) xor Cin;

end Behavioral;

**ADDSUB**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

library work;

use work.myDef.all;

entity AddSub is

Port ( A : in STD\_LOGIC\_VECTOR (n-1 downto 0);

B : in STD\_LOGIC\_VECTOR (n-1 downto 0);

Sel : in STD\_LOGIC;

Ris : out STD\_LOGIC\_VECTOR (n downto 0));

end AddSub;

architecture Behavioral of AddSub is

signal EA, EB, nB, TrueB: STD\_LOGIC\_VECTOR (n downto 0);

signal carry: STD\_LOGIC\_VECTOR (n+1 downto 0);

signal p,g: STD\_LOGIC\_VECTOR (n downto 0);

begin

EA<=A(n-1)&A; EB<=B(n-1)&B; nB<=not EB;

TrueB<=EB when Sel='0' else

nB when Sel='1'

else (others=>'X');

carry(0)<=Sel;

p<=EA xor TrueB; g<=EA and TrueB;

carry(n+1 downto 1)<=g or (p and carry(n downto 0));

Ris<=p xor carry(n downto 0);

end Behavioral;

**ADD8SRUCT**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Add8Struct is

Port ( A : in STD\_LOGIC\_VECTOR (7 downto 0);

B : in STD\_LOGIC\_VECTOR (7 downto 0);

cin : in STD\_LOGIC;

S : out STD\_LOGIC\_VECTOR (8 downto 0));

end Add8Struct;

architecture Behavioral of Add8Struct is

component FA is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

Cout : out STD\_LOGIC;

S : out STD\_LOGIC);

end component;

signal EA, EB: STD\_LOGIC\_VECTOR (8 downto 0);

signal carry: STD\_LOGIC\_VECTOR (9 downto 0);

begin

EA<=(A(7))&A; EB<=(B(7))&B;

MyFor: for i in 0 to 8 generate

FAp: FA port map (EA(i), EB(i), carry(i), carry(i+1), S(i));

end generate;

carry(0)<=cin;

end Behavioral;

**ADD8BEHA**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Add8Beh is

Port ( A : in STD\_LOGIC\_VECTOR (7 downto 0);

B : in STD\_LOGIC\_VECTOR (7 downto 0);

cin : in STD\_LOGIC;

S : out STD\_LOGIC\_VECTOR (8 downto 0));

end Add8Beh;

architecture Behavioral of Add8Beh is

signal p, g: STD\_LOGIC\_VECTOR (8 downto 0);

signal carry: STD\_LOGIC\_VECTOR (9 downto 0);

begin

p<=(A(7) xor B(7)) & (A xor B);

g<=(A(7) and B(7)) & (A and B);

carry(0)<=cin;

carry(9 downto 1)<=g or (p and carry(8 downto 0));

S<=p xor carry(8 downto 0);

end Behavioral;

**ADDNBIT**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity FA is

port(A, B, cin: in STD\_LOGIC;

cout, S: out STD\_LOGIC);

end FA;

architecture behav of FA is

begin

cout<=(A and B) or ((A xor B) and cin);

S<=A xor B xor cin;

end behav;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Add\_nbit is

generic (n:integer:=8);

Port ( A : in STD\_LOGIC\_VECTOR (n-1 downto 0);

B : in STD\_LOGIC\_VECTOR (n-1 downto 0);

cin : in STD\_LOGIC;

S : out STD\_LOGIC\_VECTOR (n downto 0));

end Add\_nbit;

architecture Behavioral of Add\_nbit is

component FA is

port(A, B, cin: in STD\_LOGIC;

cout, S: out STD\_LOGIC);

end component;

signal carry: STD\_LOGIC\_VECTOR (n+1 downto 0);

begin

myFOR: for i in 0 to n generate

myIfM: if i=n generate

FA\_M: FA port map (A(n-1), B(n-1),carry(n), carry(n+1),S(n));

end generate;

myIfL: if i<n generate

FA\_L: FA port map (A(i), B(i), carry(i), carry(i+1), S(i));

end generate;

end generate;

carry(0)<=cin;

end Behavioral;

**RIPPLE CARRY**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity rc\_adder is

port ( a : in std\_logic\_vector(7 downto 0);

b : in std\_logic\_vector(7 downto 0);

cin : in std\_logic;

sum : out std\_logic\_vector(7 downto 0);

cout : out std\_logic);

end rc\_adder;

architecture structural of rc\_adder is

component full\_adder\_con is

port (

a, b, c : in std\_logic; -- inputs

sum, ca : out std\_logic); -- sum & carry

end component;

signal carry : std\_logic\_vector(6 downto 0);

begin

U1 : full\_adder\_con port map (a(0),b(0),cin,sum(0),carry(0));

U2 : for i in 1 to 6 generate

U3 : full\_adder\_con port map (a(i),b(i),carry(i-1),sum(i),carry(i));

end generate;

U4 : full\_adder\_con port map (a(7),b(7),carry(6),sum(7),cout);

end structural;

**Adder.vhd**  
library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
use IEEE.STD\_LOGIC\_SIGNED.ALL;  
entity Adder is  
generic (nb: integer);  
Port ( A : in STD\_LOGIC\_VECTOR (nb-1 downto 0);  
B : in STD\_LOGIC\_VECTOR (nb-1 downto 0);  
Sum : out STD\_LOGIC\_VECTOR (nb downto 0));  
end Adder;  
architecture Behavioral of Adder is  
begin  
Sum<=(A(nb-1)&A)+(B(nb-1)&B);  
end Behavioral;

**Registro.vhd**

library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
entity Registro is  
generic (nb: integer);  
Port ( D : in STD\_LOGIC\_VECTOR (nb-1 downto 0);  
clk : in STD\_LOGIC;  
Q : out STD\_LOGIC\_VECTOR (nb-1 downto 0));  
end Registro;  
architecture Behavioral of Registro is  
begin  
process(clk)  
begin  
if rising\_edge(clk) then  
Q<=D;  
end if;  
end process;  
end Behavioral;

**RegAdder.vhd**  
library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
library work;  
use work.costanti.all;  
entity RegAdder is  
generic (nb:integer:=nbitin);  
Port ( A : in STD\_LOGIC\_VECTOR (nb-1 downto 0);  
B : in STD\_LOGIC\_VECTOR (nb-1 downto 0);  
clk : in STD\_LOGIC;  
RSum : out STD\_LOGIC\_VECTOR (nb downto 0));  
end RegAdder;  
architecture Behavioral of RegAdder is  
component Registro is  
generic (nb: integer);  
Port ( D : in STD\_LOGIC\_VECTOR (nb-1 downto 0);  
clk : in STD\_LOGIC;  
Q : out STD\_LOGIC\_VECTOR (nb-1 downto 0));  
end component;  
component Adder is  
generic (nb: integer);  
Port ( A : in STD\_LOGIC\_VECTOR (nb-1 downto 0);  
B : in STD\_LOGIC\_VECTOR (nb-1 downto 0);  
Sum : out STD\_LOGIC\_VECTOR (nb downto 0));  
end component;  
signal RA, RB: STD\_LOGIC\_VECTOR (nb-1 downto 0);  
signal Sum: STD\_LOGIC\_VECTOR (nb downto 0);  
begin  
RegA: Registro generic map (nb) port map (A,clk,RA);  
RegB: Registro generic map (nb) port map (B,clk,RB);  
Add: Adder generic map (nb) port map (RA,RB,Sum);  
RegS: Registro generic map (nb+1) port map (Sum,clk,RSum);  
end Behavioral

**Partial FA**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Partial\_Full\_Adder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S : out STD\_LOGIC;

P : out STD\_LOGIC;

G : out STD\_LOGIC);

end Partial\_Full\_Adder;

architecture Behavioral of Partial\_Full\_Adder is

begin

S <= A xor B xor Cin;

P <= A xor B;

G <= A and B;

end Behavioral;

|  |
| --- |
|  |

**Carry Look Ahead Adder**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Carry\_Look\_Ahead is

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

B : in STD\_LOGIC\_VECTOR (3 downto 0);

Cin : in STD\_LOGIC;

S : out STD\_LOGIC\_VECTOR (3 downto 0);

Cout : out STD\_LOGIC);

end Carry\_Look\_Ahead;

architecture Behavioral of Carry\_Look\_Ahead is

component Partial\_Full\_Adder

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S : out STD\_LOGIC;

P : out STD\_LOGIC;

G : out STD\_LOGIC);

end component;

signal c1,c2,c3: STD\_LOGIC;

signal P,G: STD\_LOGIC\_VECTOR(3 downto 0);

begin

PFA1: Partial\_Full\_Adder port map( A(0), B(0), Cin, S(0), P(0), G(0));

PFA2: Partial\_Full\_Adder port map( A(1), B(1), c1, S(1), P(1), G(1));

PFA3: Partial\_Full\_Adder port map( A(2), B(2), c2, S(2), P(2), G(2));

PFA4: Partial\_Full\_Adder port map( A(3), B(3), c3, S(3), P(3), G(3));

c1 <= G(0) OR (P(0) AND Cin);

c2 <= G(1) OR (P(1) AND G(0)) OR (P(1) AND P(0) AND Cin);

c3 <= G(2) OR (P(2) AND G(1)) OR (P(2) AND P(1) AND G(0)) OR (P(2) AND P(1) AND P(0) AND Cin);

Cout <= G(3) OR (P(3) AND G(2)) OR (P(3) AND P(2) AND G(1)) OR (P(3) AND P(2) AND P(1) AND G(0)) OR (P(3) AND P(2) AND P(1) AND P(0) AND Cin);

end Behavioral;