**Half Adder**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

library UNISIM;

use UNISIM.VComponents.all;

entity Half\_adder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end Half\_adder;

architecture Behavioral of Half\_adder is

begin

S <= A XOR B;

C <= A AND B;

end Behavioral;

**Half Adder testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

library UNISIM;

use UNISIM.VComponents.all;

entity Half\_adder\_tb is

end Half\_adder\_tb;

architecture Behavioral of Half\_adder\_tb is

signal A0 : std\_logic;

signal B0 : std\_logic;

signal S0 : std\_logic;

signal C0 : std\_logic;

begin

p0 : entity work.Half\_adder(behavioral) port map (A => A0, B => B0, S => S0, C => C0);

process

begin

wait for 100ns;

A0 <= '1';

B0 <= '0';

wait for 100ns;

A0 <= '1';

B0 <= '1';

wait for 100ns;

A0 <= '0';

B0 <= '0';

wait for 100ns;

A0 <= '0';

B0 <= '1';

wait for 100ns;

end process;

end Behavioral;

A screenshot of a computer

Description automatically generated

**Half adder simulation**

**Full Adder**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

use work.all;

library UNISIM;

use UNISIM.VComponents.all;

entity Full\_adder is

Port ( A1 : in STD\_LOGIC;

B1 : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S1 : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end Full\_adder;

architecture Behavioral of Full\_adder is

component Half\_adder

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end component Half\_adder;

for all : Half\_adder use entity work.Half\_adder(behavioral);

signal ha1out1 : std\_logic;

signal ha1out2 : std\_logic;

signal ha2in1 : std\_logic;

signal ha2out2 : std\_logic;

begin

ha1 : Half\_adder port map(A1, B1, ha1out1, ha1out2);

ha2 : Half\_adder port map(ha2in1, Cin, S1, ha2out2);

ha2in1 <= ha1out1;

Cout <= ha1out2 OR ha2out2;

end Behavioral;

**Full Adder testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

library UNISIM;

use UNISIM.VComponents.all;

entity Full\_adder\_tb is

end Full\_adder\_tb;

architecture Behavioral of Full\_adder\_tb is

signal A0 : std\_logic;

signal B0 : std\_logic;

signal Cin0 : std\_logic;

signal S0 : std\_logic;

signal Cout0 : std\_logic;

begin

p1 : entity work.Full\_adder(behavioral) port map(A0, B0, Cin0, S0, Cout0);

process

begin

wait for 100ns;

A0 <= '0';

B0 <= '0';

Cin0 <= '0';

wait for 100ns;

A0 <= '0';

B0 <= '0';

Cin0 <= '1';

wait for 100ns;

A0 <= '0';

B0 <= '1';

Cin0 <= '0';

wait for 100ns;

A0 <= '0';

B0 <= '1';

Cin0 <= '1';

wait for 100ns;

A0 <= '1';

B0 <= '0';

Cin0 <= '0';

wait for 100ns;

A0 <= '1';

B0 <= '0';

Cin0 <= '1';

wait for 100ns;

A0 <= '1';

B0 <= '1';

Cin0 <= '0';

wait for 100ns;

A0 <= '1';

B0 <= '1';

Cin0 <= '1';

wait for 100ns;

end process;

end Behavioral;

Graphical user interface, application

Description automatically generated

**Full adder simulation**

**Ripple Carry Adder**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

use work.all;

library UNISIM;

use UNISIM.VComponents.all;

entity Ripple\_carry\_adder is

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

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

CinR : in STD\_LOGIC;

SR : out STD\_LOGIC\_VECTOR (7 downto 0);

CoutR : out STD\_LOGIC);

end Ripple\_carry\_adder;

architecture Behavioral of Ripple\_carry\_adder is

component Half\_adder

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end component Half\_adder;

for all : Half\_adder use entity work.Half\_adder(behavioral);

component Full\_adder

Port ( A1 : in STD\_LOGIC;

B1 : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S1 : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end component Full\_adder;

for all : Full\_adder use entity work.Full\_adder(behavioral);

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

signal carryOut : std\_logic\_vector(7 downto 0);

begin

ha : Half\_adder port map(AR(0), BR(0), SR(0), carryOut(0));

gen\_fa : for I in 1 to 7 generate

carryIn(I-1) <= carryOut(I-1);

faX : Full\_adder port map(AR(I), BR(I), carryIn(I-1), SR(I), carryOut(I));

end generate gen\_fa;

CoutR <= carryOut(7);

end Behavioral;

**Ripple Carry Adder testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

library UNISIM;

use UNISIM.VComponents.all;

entity Ripple\_carry\_adder\_tb is

end Ripple\_carry\_adder\_tb;

architecture Behavioral of Ripple\_carry\_adder\_tb is

signal AR : std\_logic\_vector(7 downto 0);

signal BR : std\_logic\_vector(7 downto 0);

signal CinR : std\_logic;

signal SR : std\_logic\_vector(7 downto 0);

signal CoutR : std\_logic;

begin

p2 : entity work.Ripple\_carry\_adder(behavioral) port map(AR, BR, CinR, SR, CoutR);

process

begin

CinR <= '0';

wait for 100ns;

AR <= "01001001";

BR <= "01001001";

wait for 100ns;

AR <= "10000001";

BR <= "10011001";

wait for 100ns;

AR <= "10111111";

BR <= "00000001";

wait for 100ns;

AR <= "01010101";

BR <= "10101010";

wait for 100ns;

AR <= "11111111";

BR <= "00000001";

wait for 100ns;

end process;

end Behavioral;

A screenshot of a computer

Description automatically generated with medium confidence

**Ripple carry adder simulation**