

**آزمایش شماره ۴**

**Half Adder:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Half\_Adder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Sum : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end Half\_Adder;

architecture Gate\_Level of Half\_Adder is

begin

Sum <= A XOR B;

Cout <= A AND B;

end Gate\_Level;

**Full Adder:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Full\_Adder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

Sum : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end Full\_Adder;

architecture Gate\_Level of Full\_Adder is

begin

Sum <= (A XOR B) XOR Cin;

Cout <= (A AND B) OR (Cin AND (A XOR B));

end Gate\_Level;

**Ripple Adder:**

entity Ripple\_Adder is

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

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

Cin : in STD\_LOGIC;

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

Cout : out STD\_LOGIC);

end Ripple\_Adder;

architecture Behavioral of Ripple\_Adder is

component Full\_Adder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

Sum : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end component;

signal carry : std\_logic\_vector(4 downto 0);

begin

FA0: Full\_Adder port map (A(0), B(0), Cin, Sum(0), carry(0));

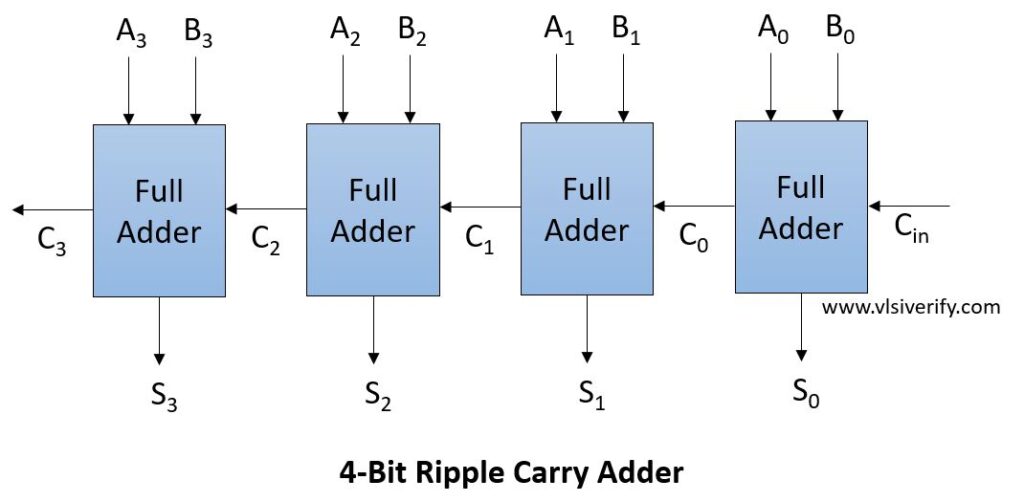
FA1: Full\_Adder port map (A(1), B(1), carry(0), Sum(1), carry(1));

FA2: Full\_Adder port map (A(2), B(2), carry(1), Sum(2), carry(2));

FA3: Full\_Adder port map (A(3), B(3), carry(2), Sum(3), Cout);

end Behavioral;

جمع کننده Ripple Adderبا اضافه کردن هر بیت به صورت موازی و انتشار انتقال از بیت های مرتبه پایین به بیت های مرتبه بالاتر، جمع دودویی را انجام می دهد. انجام از هر مرحله تبدیل به Carryدر مرحله بعدی می شود که در انجام نهایی که سرریز را نشان می دهد به اوج می رسد. تاخیر انتشار آن با تعداد بیت ها افزایش می یابد و کارایی آن را برای اعداد بزرگ کاهش می دهد.



**Simulation**:



**BCD Adder:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity BCDb\_Ripple is

Port(

Abcd, Bbcd: in std\_logic\_vector(3 downto 0);

Sumbcd : out std\_logic\_vector(3 downto 0);

Coutbcd : out std\_logic

);

end BCDb\_Ripple;

architecture Behavioral of BCDb\_Ripple is

component Ripple\_Adder4bit is

Port(

A, B: in std\_logic\_vector(3 downto 0);

Sum : out std\_logic\_vector(3 downto 0);

Cout : out std\_logic

);

end component Ripple\_Adder4bit;

signal cin1, cin2, cin3, cin\_last, coutRipple1, coutRipple2, and\_11xx, and\_1x1x, or\_3, xor\_s3\_s3prime : std\_logic;

signal Ssum, Bfor0110 : std\_logic\_vector(3 downto 0);

begin

adder\_instance0: Ripple\_Adder4bit port map(A => Abcd, B => Bbcd, Sum => Ssum, Cout => coutRipple1);

and\_11xx <= Ssum(3) and Ssum(2);

and\_1x1x <= Ssum(3) and Ssum(1);

or\_3 <= and\_11xx or and\_1x1x or coutRipple1;

Bfor0110(0) <= '0';

Bfor0110(1) <= or\_3;

Bfor0110(2) <= or\_3;

Bfor0110(3) <= '0';

adder\_instance1: Ripple\_Adder4bit port map(A => Ssum, B => Bfor0110, Sum => Sumbcd, Cout => coutRipple2);

Coutbcd <= or\_3 or coutRipple2;

end Behavioral;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Ripple\_Adder4bit is

Port(

A, B: in std\_logic\_vector(3 downto 0);

Sum : out std\_logic\_vector(3 downto 0);

Cout : out std\_logic

);

end Ripple\_Adder4bit;

architecture structure of Ripple\_Adder4bit is

component FullAdder is

Port(

a, b, cin: in std\_logic;

sum, cout: out std\_logic

);

end component FullAdder;

signal cin1, cin2, cin3: std\_logic;

begin

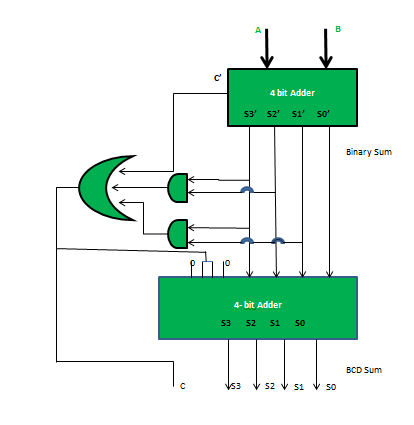
fullAdder0: FullAdder port map(a => A(0), b => B(0), cin => '0', sum => Sum(0), cout => cin1);

fullAdder1: FullAdder port map(a => A(1), b => B(1), cin => cin1, sum => Sum(1), cout => cin2);

fullAdder2: FullAdder port map(a => A(2), b => B(2), cin => cin2, sum => Sum(2), cout => cin3);

fullAdder3: FullAdder port map(a => A(3), b => B(3), cin => cin3, sum => Sum(3), cout => Cout);

end structure;



جمع کننده BCD یک مدار دیجیتالی است که دو عدد اعشاری کد شده باینری (BCD) را اضافه می کند. BCD روشی برای نمایش ارقام اعشاری به شکل دودویی است که معمولاً از چهار بیت برای نمایش هر رقم استفاده می کند. برای پیاده سازی یک جمع کننده BCD با استفاده از دو جمع کننده 4 بیتی، فرآیند افزودن BCD را به دو مرحله تقسیم می کنیم:

جمع دودویی: ابتدا دو عدد BCD را به عنوان اعداد باینری با استفاده از دو جمع کننده 4 بیتی اضافه می کنیم. این مرحله یک مجموع باینری را به همراه یک عمل انجام می دهد.

تنظیم BCD: پس از جمع دودویی، نتیجه را تنظیم می کنیم تا مطمئن شویم که در محدوده BCD (0 تا 9 برای هر رقم) باقی می ماند. این تنظیم شامل تصحیح مجموع دودویی بیشتر از 9 یا هر گونه انتقال بین ارقام BCD است.

با ترکیب این مراحل، می توانیم به طور موثر دو عدد BCD را با استفاده از دو جمع کننده 4 بیتی اضافه کنیم و نتیجه BCD درستی را تولید کنیم.

**Simulation:**

