اهداف آزمایش: در این آزمایش قصد داریم بهینه سازی در طراحی جمع کننده ها انجام دهیم. اینکار را برای RA و CSA و CLA انجام دهیم و درنهایت تاخیر و هزینه آن را مورد بررسی قرار دهیم.

جمع کننده RA:

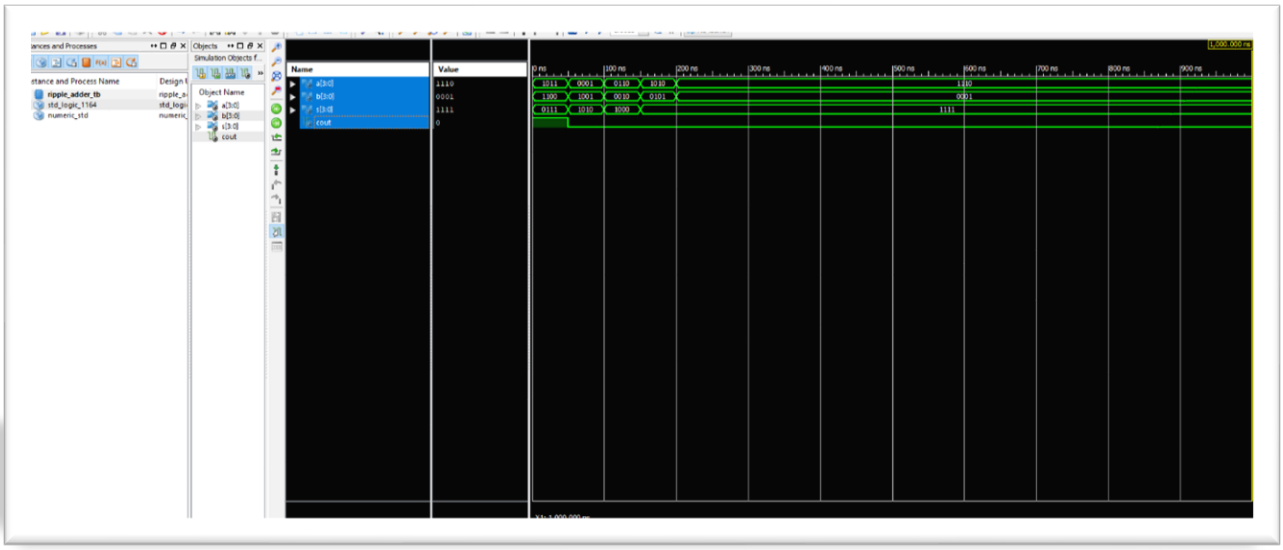
برای این مدل جمع کننده به جای اولین FA یک عدد HA قرار میدهیم. دقت شود اینکار سبب آن میشود که کل جمع کننده (بلک باکس)، Cin نداشته باشد. کد آن به شکل زیر خواهد بود:

Ripple_Adder.vhd

```
entity Ripple_Adder is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
    B : in STD_LOGIC_VECTOR (3 downto 0);
    S : out STD_LOGIC_VECTOR (3 downto 0);
    Cout : out STD_LOGIC);
end Ripple_Adder;

architecture Behavioral of Ripple_Adder is

 -- Full Adder VHDL Code Component Decalaration
     component full_adder
        Port (A : in STD_LOGIC;
                B : in STD_LOGIC;
                Cin : in STD_LOGIC;
                sum : out STD_LOGIC;
                Cout : out STD_LOGIC);
     end component full_adder;

     component half_adder
        Port (A : in STD_LOGIC;
                B : in STD_LOGIC;
                sum : out STD_LOGIC;
                Cout : out STD_LOGIC);
     end component half_adder;

signal c1,c2,c3: STD_LOGIC;

begin

    -- Port Mapping Full Adder 4 times
    HA1: half_adder port map( A(0), B(0), S(0), c1);
    FA2: full_adder port map( A(1), B(1), c1, S(1), c2);
    FA3: full_adder port map( A(2), B(2), c2, S(2), c3);
    FA4: full_adder port map( A(3), B(3), c3, S(3), Cout);

end Behavioral;
```

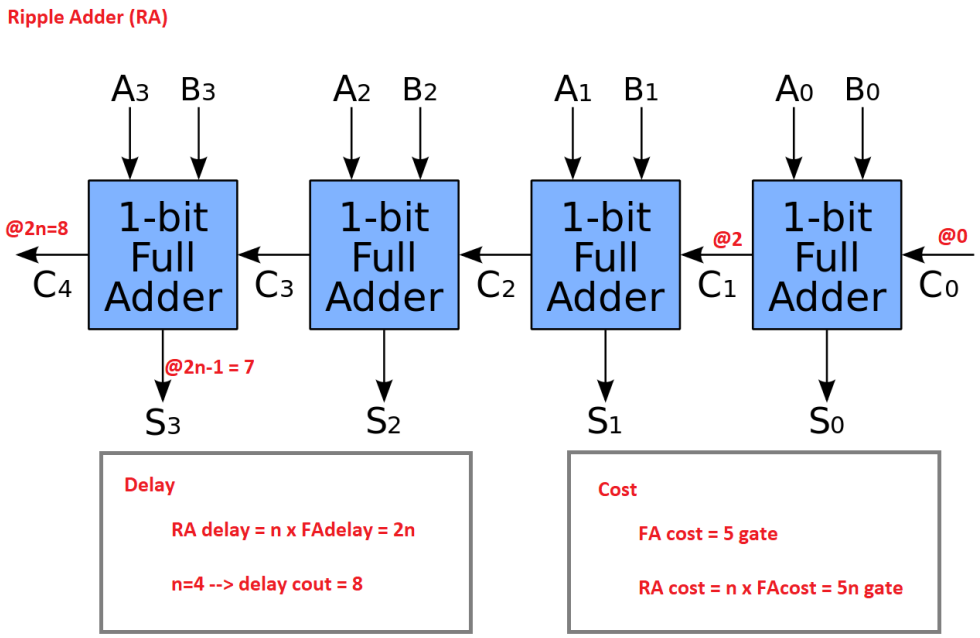
همانطور که از کد فوق پیداست، به جای FA1 ، یک عدد HA قرار داده ایم.

شکل موج آن به شکل زیر است:



براساس شکل موج و مدل طراحی، تحلیل زیر حاصل میشود:

البته این نکته حائز اهمیت است که FA اول با یک HA جایگزین میشود و از آنجایی که Cost، Half Adder معادل دو گیت است و به همین ترتیب Delay نیز متفاوت است.



جمع کننده CLA:

براساس آنچه در جلسه قبل درمورد CLA بررسی کردیم، اقدام به طراحی این مدل جمع کننده میکنیم. در اینجا ابتدا دو ماژول Partial_Half_Adder و Partial_Full_Adder را تعریف میکنیم:

Partial_Full_Adder.vhd

```
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;
```

Partial_Half_Adder.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Partial_Half_Adder is
    Port (A : in STD_LOGIC;
          B : in STD_LOGIC;
          S : out STD_LOGIC;
          P : out STD_LOGIC;
          G : out STD_LOGIC);
end Partial_Half_Adder;

architecture Behavioral of Partial_Half_Adder is

begin
    S <= A xor B;
    P <= A xor B;
    G <= A and B;
end Behavioral;
```

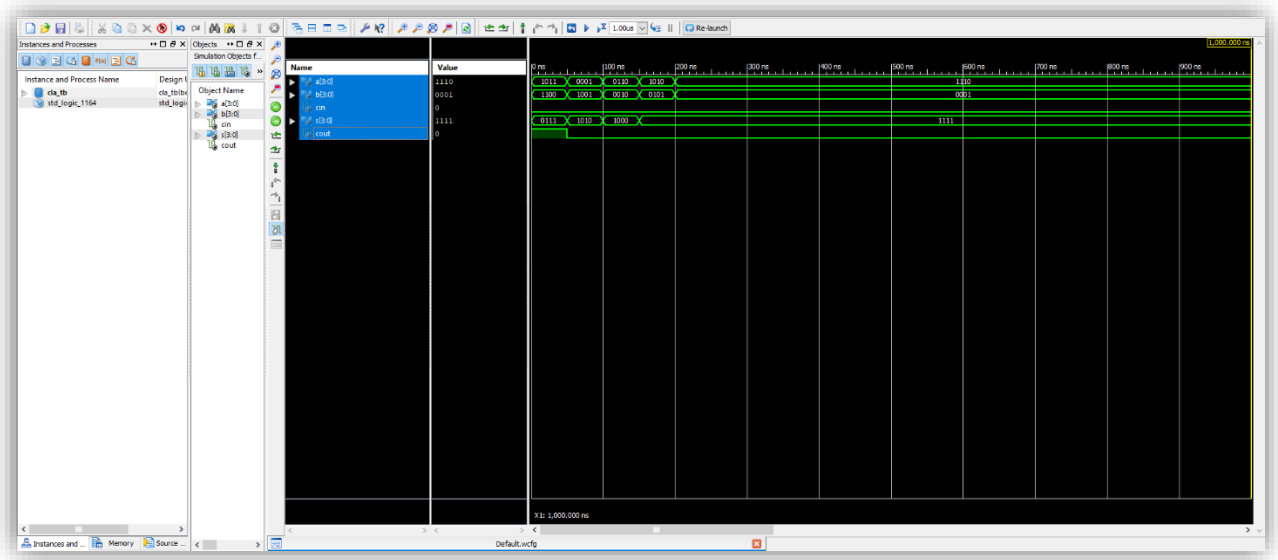
حال بر اساس این دو ماژول، ماژول اصلی را تعریف میکنیم:

CLA.vhd

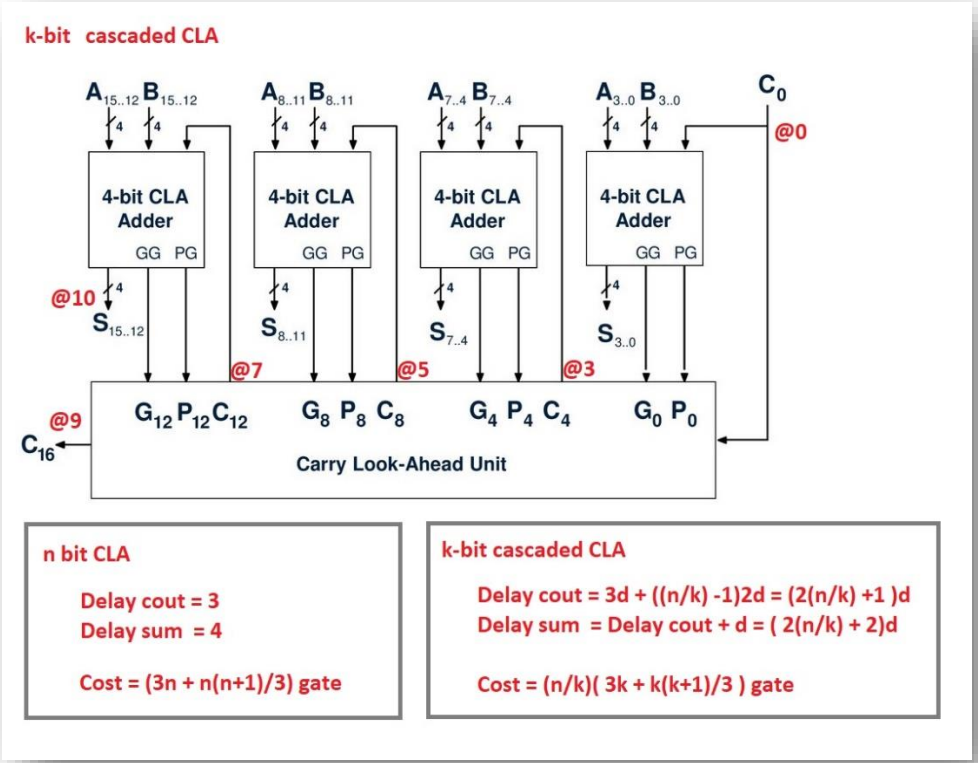
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Carry_Look_Ahead is
    Port (A : in STD_LOGIC_VECTOR (3 downto 0);
          B : in STD_LOGIC_VECTOR (3 downto 0);
          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;

    component Partial_Half_Adder
        Port (A : in STD_LOGIC;
              B : 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

    PHA1: Partial_Half_Adder port map( A(0), B(0), 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);
    c2 <= G(1) OR (P(1) AND G(0));
    c3 <= G(2) OR (P(2) AND G(1)) OR (P(2) AND P(1) AND G(0));
    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));

end Behavioral;
```

همانطور که پیداست در PHA1 از Partial Half Adder استفاده کرده ایم. نکته ای که برای RA داشتیم را اینبار برای اینجا نیز داریم. Cin ای وجود دارد بنابراین ماژول قابل cascade نیست این بدین معناست که نمیتوان از چندتا از آن برای ایجاد جمع کننده بزرگتر استفاده کرد (چون عامل کانکت شوندگی یا همان Cin را ندارند). این موضوع در تست بنچ (واقع درپیوست) نیز قابل تصدیق است.

شکل موج آن به شکل زیر است:



و تاخیر و هزینه آن را محاسبه میکنیم:



جمع کننده CSA:

اینبار میخواهیم این بهینه سازی را برای Carry Save Adder انجام دهیم. فرض کرده ایم که سه عدد چهار بیتی داریم که حاصل جمع آن را میخواهیم به علاوه نقلی خروجی!

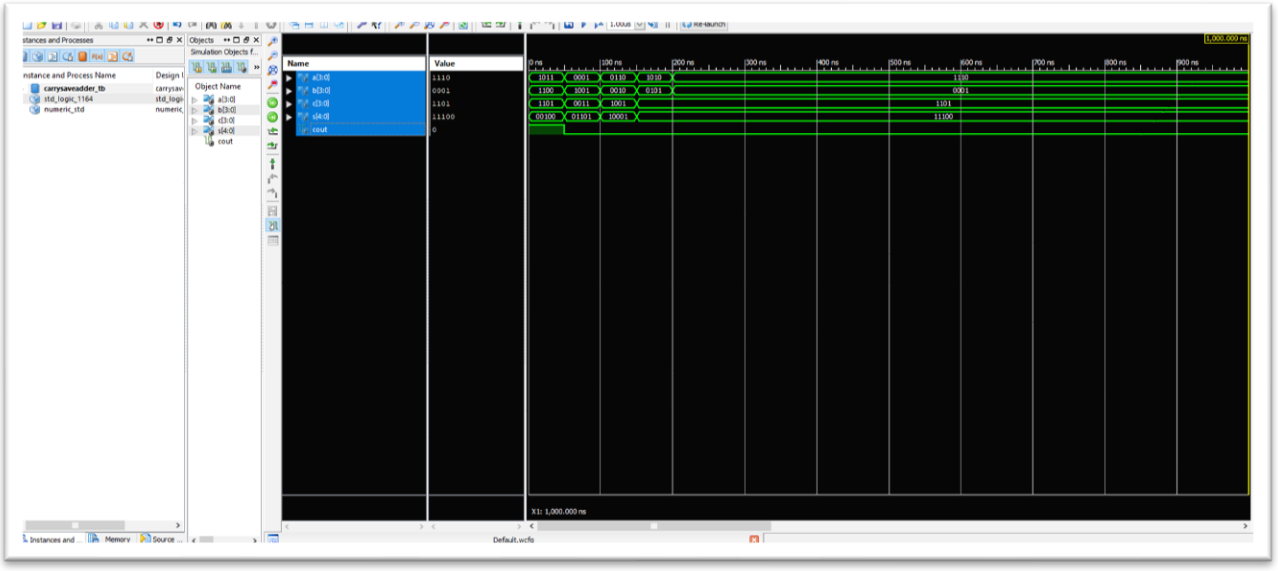
برای اینکار از ماژول های از پیش طراحی شده و استفاده شده Full Adder و Half Adder استفاده میکنیم.
(در انتهای این داکیومنت کد آن آمده است)

CSA.vhd

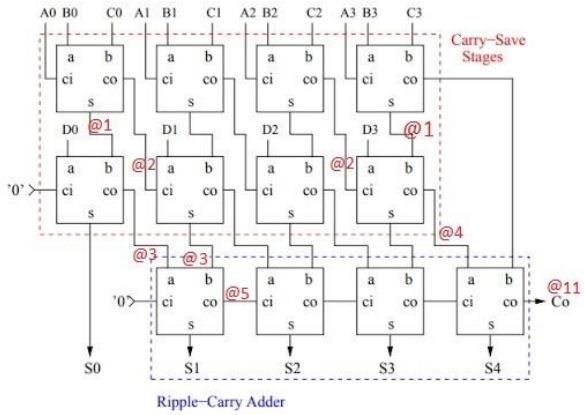
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity carry_save_adder is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
    B : in STD_LOGIC_VECTOR (3 downto 0);
    C : in STD_LOGIC_VECTOR (3 downto 0);
    S : OUT STD_LOGIC_VECTOR (4 downto 0);
    Cout : OUT STD_LOGIC);
end carry_save_adder;

architecture Behavioral of carry_save_adder is

    component full_adder
            Port (A : in STD_LOGIC;
                    B : in STD_LOGIC;
                    Cin : in STD_LOGIC;
                    sum : out STD_LOGIC;
                    Cout : out STD_LOGIC);
    end component;

    component half_adder
            Port (A : in STD_LOGIC;
                    B : in STD_LOGIC;
                    sum : out STD_LOGIC;
                    Cout : out STD_LOGIC);
    end component;

 -- Intermediate signal
signal X,Y: STD_LOGIC_VECTOR(3 downto 0);
signal C1,C2,C3: STD_LOGIC;

begin
 -- Carry save adder block
    FA1: full_adder PORT MAP(A(0),B(0),C(0),S(0),X(0));
    FA2: full_adder PORT MAP(A(1),B(1),C(1),Y(0),X(1));
    FA3: full_adder PORT MAP(A(2),B(2),C(2),Y(1),X(2));
    FA4: full_adder PORT MAP(A(3),B(3),C(3),Y(2),X(3));

     -- Ripple carry adder block
    HA5: half_adder PORT MAP(X(0),Y(0),S(1),C1);
    FA6: full_adder PORT MAP(X(1),Y(1),C1,S(2),C2);
    FA7: full_adder PORT MAP(X(2),Y(2),C2,S(3),C3);
    HA8: half_adder PORT MAP(X(3), C3, S(4), Cout);

end Behavioral;
```

همانطور که در کد فوق پیداست در دو بخش (دو قسمت) با توجه به نحوه الگوریتم جمع ذخیره گر نقلی، آمده ایم و به جای افزایش هزینه و استفاده مجدد از FA، HA استفاده کرده ایم.

شکل موج آن به صورت زیر خواهد بود:



و تحلیل آن به صورت زیر است:



Delay

Delay cout = Delay RA input + 2nd = 3 + 2x4 = 11

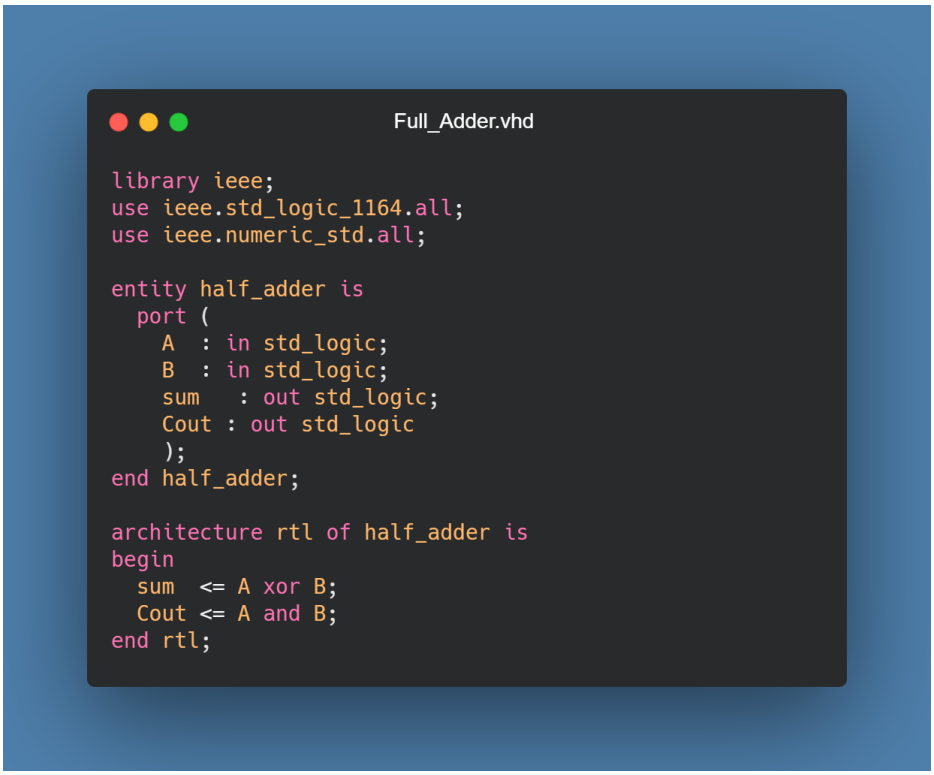
Cost = 12 cost FA = 12x5 gate = 60 gate

ماژول Full Adder:

Full_Adder.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.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 rtl of full_adder is

  signal w_WIRE_1 : std_logic;
  signal w_WIRE_2 : std_logic;
  signal w_WIRE_3 : std_logic;

begin

  w_WIRE_1 <= A xor B;
  w_WIRE_2 <= w_WIRE_1 and Cin;
  w_WIRE_3 <= A and B;

  sum  <= w_WIRE_1 xor Cin;
  Cout <= w_WIRE_2 or w_WIRE_3;


end rtl;
```

ماژول Half Adder:



تمامی فایل ها در پیوست موجود است.

موفق باشید