Computer Architecture Lab Session 5

# Ripple Adder

# Carry look Ahead Adder

# Carry Save Adder

# Bardia Ardakanian 9831072

# Ali Asad 9831004

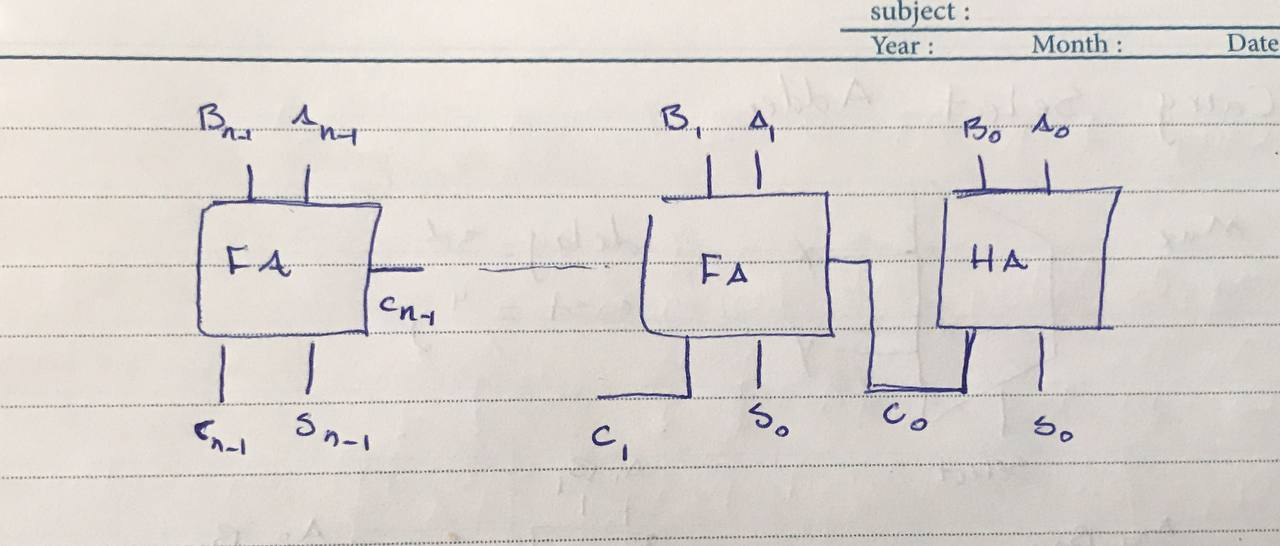
## (Ripple-Adder) جمع کننده آبشاری

یک بهینه سازی که می‌توان در این جمع کننده نسبت به ازمایش هفته قبل داشت جایگزین کردن اولین Full Adder با یک Half Adder است.

اگر شکل زیر دیاگرام یک Ripple Adder کامل باشد:



میتوان با حذف کردن FA اول و جایگزین کردن آن با HA هزینه را کاهش داد، به صورت زیر:



چون در RA ما Carry in نداریم میتوان این کار را انجام داد و همجنان نتیجه جمع کننده درست باقی بماند.

کد vhdl این RA 4 بیتی را باهم مشاهده می‌کنیم:

|  |
| --- |
| library IEEE; use IEEE.STD\_LOGIC\_1164.ALL;  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;  S : out STD\_LOGIC\_VECTOR (3 downto 0);  Cout : out STD\_LOGIC); end Ripple\_Adder;  architecture Behavioral of Ripple\_Adder is  component FA Port (   A : in STD\_LOGIC;  B : in STD\_LOGIC;  Cin : in STD\_LOGIC;  S : out STD\_LOGIC;  Cout : out STD\_LOGIC); end component;  component HA Port (   A : in STD\_LOGIC;  B : in STD\_LOGIC;  S : out STD\_LOGIC;  Cout : out STD\_LOGIC); end component;   -- Intermediate Carry declaration signal c1,c2,c3: STD\_LOGIC; begin   -- Port Mapping Full Adder 4 times  HA1: HA port map( A(0), B(0), S(0), c1);  FA1: FA port map( A(1), B(1), c1, S(1), c2);  FA2: FA port map( A(2), B(2), c2, S(2), c3);  FA3: FA port map( A(3), B(3), c3, S(3), Cout);  end Behavioral; |

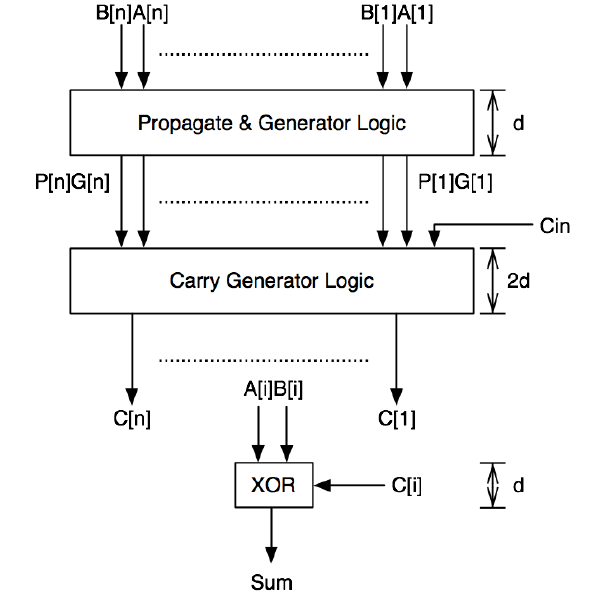
کد Test Bench

|  |
| --- |
| LIBRARY ieee; USE ieee.std\_logic\_1164.ALL;  ENTITY Tb\_Ripple\_Adder IS END Tb\_Ripple\_Adder;  ARCHITECTURE behavior OF Tb\_Ripple\_Adder IS   -- Component Declaration for the Unit Under Test (UUT)  COMPONENT Ripple\_Adder 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 COMPONENT;  --Inputs signal A : std\_logic\_vector(3 downto 0) := (others => '0'); signal B : std\_logic\_vector(3 downto 0) := (others => '0'); signal Cin : std\_logic := '0';  --Outputs signal S : std\_logic\_vector(3 downto 0); signal Cout : std\_logic;  BEGIN   -- Instantiate the Unit Under Test (UUT) uut: Ripple\_Adder PORT MAP (  A => A,  B => B,  Cin => Cin,  S => S,  Cout => Cout );   -- Stimulus process stim\_proc: process begin  -- hold reset state for 100 ns.  wait for 100 ns;  A <= "0110";  B <= "1100";   wait for 100 ns;  A <= "1111";  B <= "1100";   wait for 100 ns;  A <= "0110";  B <= "0111";   wait for 100 ns;  A <= "0110";  B <= "1110";   wait for 100 ns;  A <= "1111";  B <= "1111";   wait;  end process;  END; |

## جمع کننده با پیش‌بینی بیت نقلی (Carry Look-ahead Adder)

همانطور که در ازمایش قبل گفتیم این جمع کننده چون Carry ها را پیش بینی می‌کند از Pi ها و Gi ها و sop از انها استفاده می‌کند. مانند RA نمیشود پیزی از ان کم کرد و در بهینه ترین شرایط ممکن است.

دیاگرام CSA



کد vhdl این جمع کننده را در حالت 4 بیتی مشاهده می‌کنیم:

کد Partial Adder:

|  |
| --- |
| 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; |

کد CSA

|  |
| --- |
| 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; |

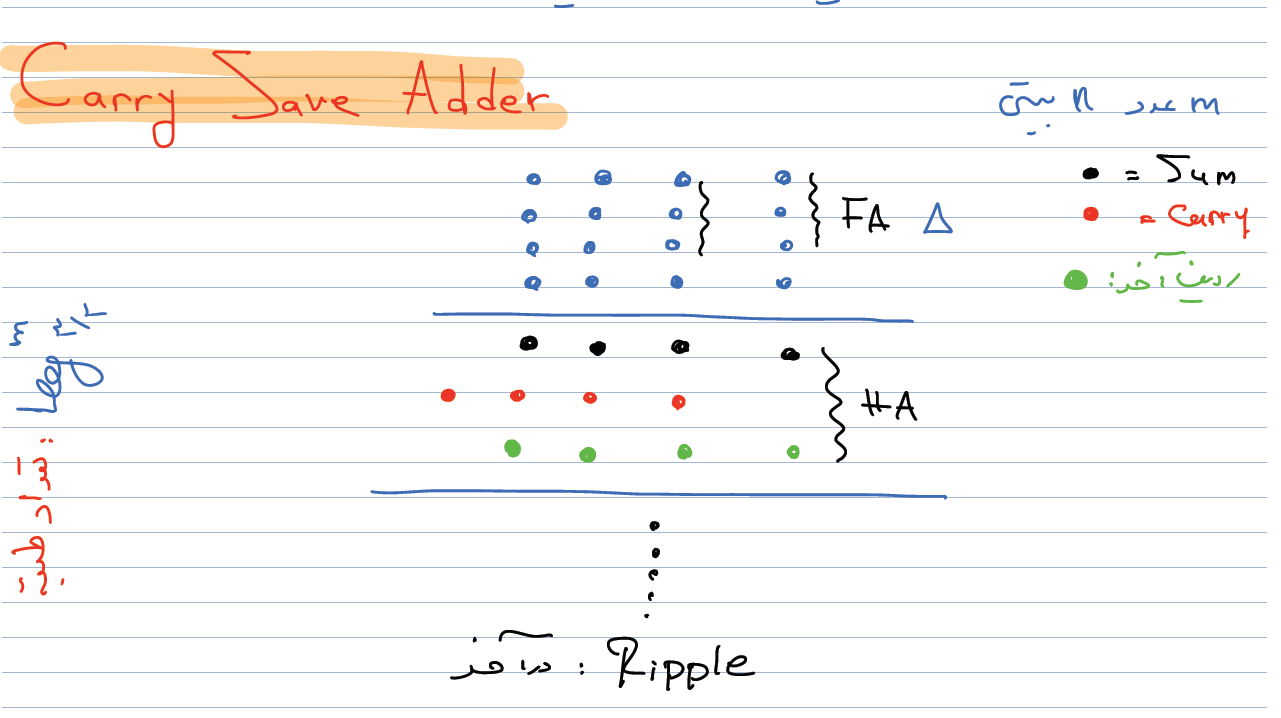
کد Test Bench

|  |
| --- |
| LIBRARY ieee; USE ieee.std\_logic\_1164.ALL;  ENTITY Tb\_Carry\_Look\_Ahead IS END Tb\_Carry\_Look\_Ahead;  ARCHITECTURE behavior OF Tb\_Carry\_Look\_Ahead IS   -- Component Declaration for the Unit Under Test (UUT) COMPONENT Carry\_Look\_Ahead 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 COMPONENT;  --Inputs signal A : std\_logic\_vector(3 downto 0) := (others => '0'); signal B : std\_logic\_vector(3 downto 0) := (others => '0'); signal Cin : std\_logic := '0';  --Outputs signal S : std\_logic\_vector(3 downto 0); signal Cout : std\_logic;  BEGIN   -- Instantiate the Unit Under Test (UUT) uut: Carry\_Look\_Ahead PORT MAP (  A => A,  B => B,  Cin => Cin,  S => S,  Cout => Cout );   -- Stimulus process stim\_proc: process begin  -- hold reset state for 100 ns.  wait for 10 ns;   A <= "1111";  B <= "1111";  Cin <= '1';   wait for 10 ns;   A <= "1010";  B <= "0111";  Cin <= '0';   wait for 10 ns;   A <= "1000";  B <= "1001";  Cin <= '0';   wait;  end process;  END; |

## جمع کننده ذخیره گر (Carry Save Adder)

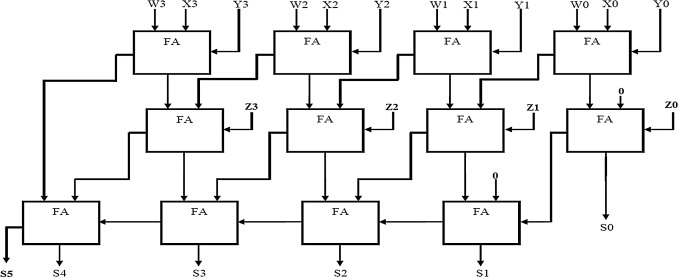
این جمع کننده با مابقی جمع کننده هایی که قبلا راجبشان صحبت کرده ایم متفاوت است. چرا که توانایی جمع کردن بیش از 2 عدد را دارد.

همانطور که قبلا گفتیم یک FA می‌تواند 3 بیت را باهم جمع بزند به صورتی که عدد اول A و عدد درم B و عدد سوم به عنوان Carry. از این خاصت FA استفاده می‌کنیم تا یک جمع کننده برای جمع زدن بیش از 2 عدد را بسازیم. روش کلی این جمع کننده به صورت زیر است:

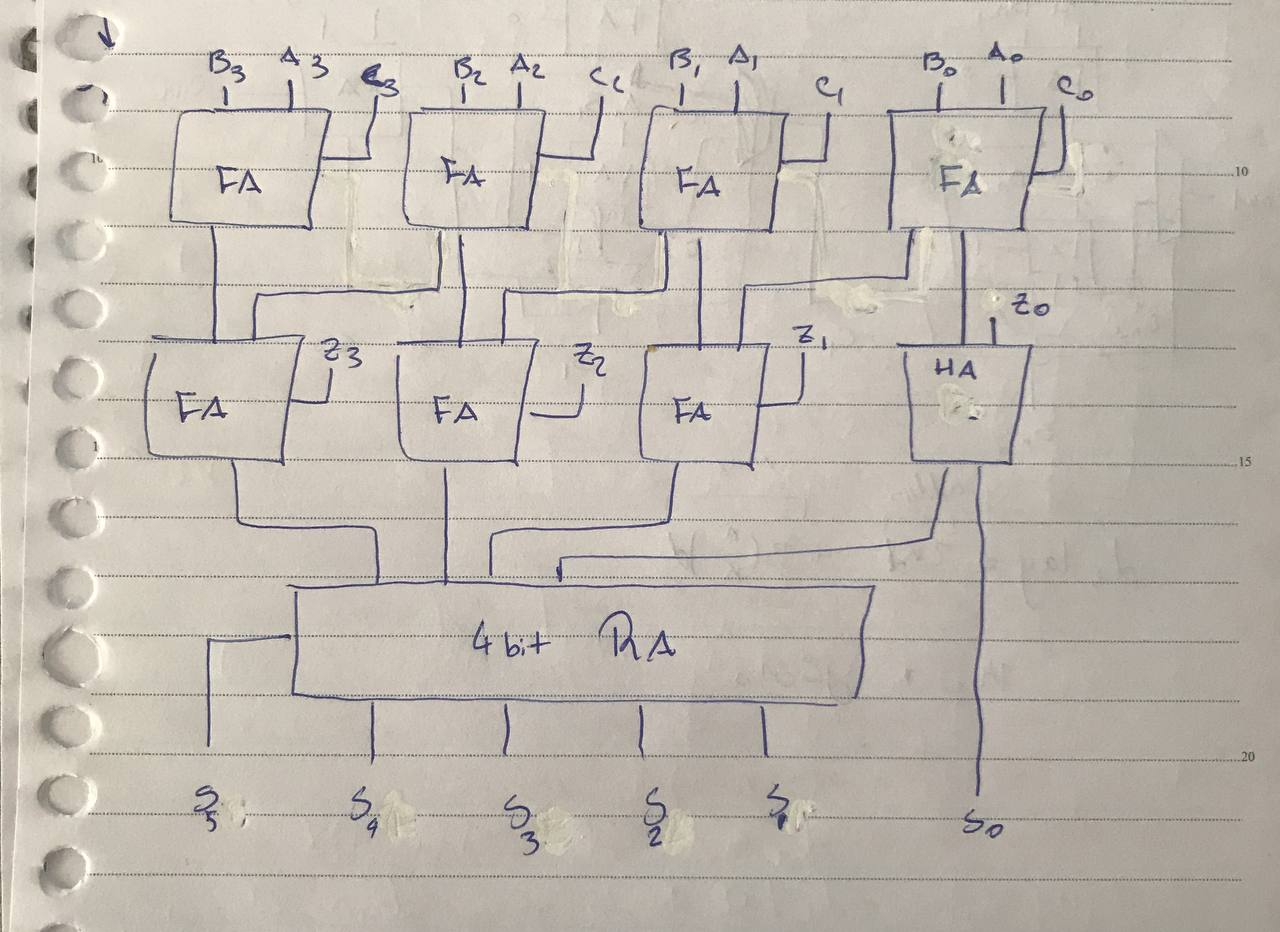


کاری که می‌کند این است 3 تا 3 تا جمع می‌زند و مابقی را در ردیف بعدی به همین ترتیب جمع میزند. در هر مرحله عدد نتیجه 1 بیت بزرگ تر می‌شود(بخاطر carry). این روند را انقدر ادامه میدهیم تا 2 عدد به دست بیاید و در انتها این 2 عدد را با استفاده ار Ripple Adder جمع می‌زنیم.

دیاگرام این جمع کننده به صورت زیر است:



در این مثال هم می‌توان FA اولی را با HA جایگزین کرد. همچنین ‌می‌توان مانند مثال این گزارش از Ripple Adder بهینه تری استفاده شود. به صورت زیر:



کد این جمع کننده به صورت زیر است:

|  |
| --- |
| 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 FA Port (   A : in STD\_LOGIC;  B : in STD\_LOGIC;  Cin : in STD\_LOGIC;  S : out STD\_LOGIC;  Cout : out STD\_LOGIC); end component;  component HA Port (   A : in STD\_LOGIC;  B : in STD\_LOGIC;  S : 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  HA1: HA PORT MAP(A(0),B(0),S(0),X(0));  FA2: FA PORT MAP(A(1),B(1),C(1),Y(0),X(1));  FA3: FA PORT MAP(A(2),B(2),C(2),Y(1),X(2));  FA4: FA PORT MAP(A(3),B(3),C(3),Y(2),X(3));   -- Ripple carry adder block  HA5: HA PORT MAP(X(0),Y(0),S(1),C1);  FA6: FA PORT MAP(X(1),Y(1),C1,S(2),C2);  FA7: FA PORT MAP(X(2),Y(2),C2,S(3),C3);  FA8: FA PORT MAP(X(3),'0',C3,S(4),Cout);  end Behavioral; |

کد تست بنچ:

|  |
| --- |
| LIBRARY ieee; USE ieee.std\_logic\_1164.ALL;  ENTITY Tb\_carry\_save IS END Tb\_carry\_save;  ARCHITECTURE behavior OF Tb\_carry\_save IS   -- Component Declaration for the Unit Under Test (UUT)  COMPONENT carry\_save\_adder 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 COMPONENT;  --Inputs signal A : std\_logic\_vector(3 downto 0) := (others => '0'); signal B : std\_logic\_vector(3 downto 0) := (others => '0'); signal C : std\_logic\_vector(3 downto 0) := (others => '0');  --Outputs signal S : std\_logic\_vector(4 downto 0); signal Cout : std\_logic;  BEGIN   -- Instantiate the Unit Under Test (UUT) uut: carry\_save\_adder PORT MAP (  A => A,  B => B,  C => C,  S => S,  Cout => Cout );   -- Stimulus process stim\_proc: process begin  -- hold reset state for 100 ns.  wait for 100 ns;  A <= "1100";  B <= "1101";  C <= "1110";   wait for 100 ns;  A <= "1111";  B <= "1000";  C <= "1001";   wait for 100 ns;  A <= "1110";  B <= "0101";  C <= "0111";   wait;  end process;  END; |

هزینه این از هزینه CSA اولیه کمی کمتر است.