دانشکده مهندسی کامپیوتر

آزمایشگاه معماری کامپیوتر - گزارش کار آزمایش ۲

استاد : خانم دکتر پریا دربانی

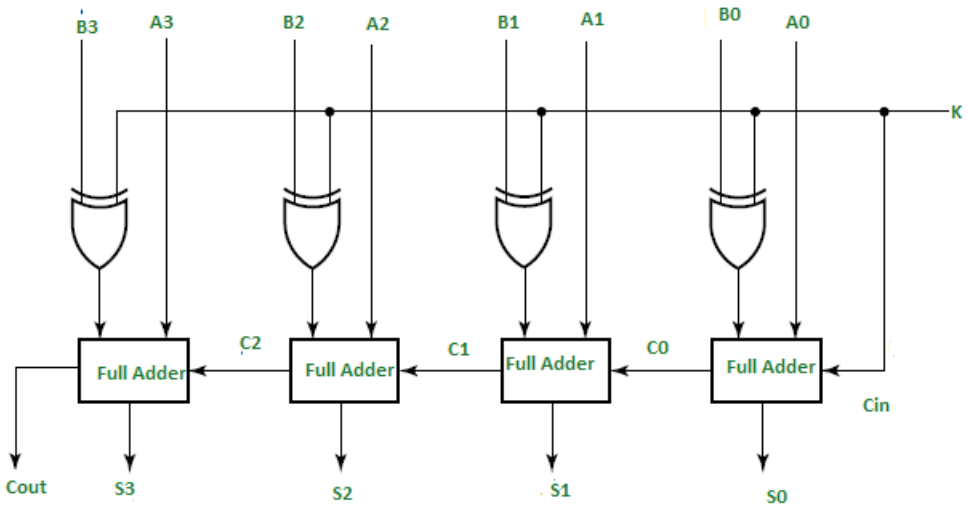
سید مهدی رضوی

پاییز ۱۴۰۱

فهرست مطالب

۱ مقدمه

در این آزمایش به بررسی مدار سخت افزاری یک تمام جمع کننده چهار بیتی (Full Adder 4 bit) می پردازیم .
برای این کار مطابق شکل ۱ نیاز به چهار قطعه تمام جمع کننده Full Adder خواهیم داشت.
سپس یک قطعه سخت افزاری تمام جمع کننده را به کمک دو قطعه نیم جمع کننده Half Adder پیاده سازی خواهیم کرد.
در نهایت نیز با نوشتن تست قطعات Components را آزمایش خواهیم کرد.



شکل ۱: شمای یک جمع کننده چهار بیتی

۲ تمام جمع کننده FullAdder

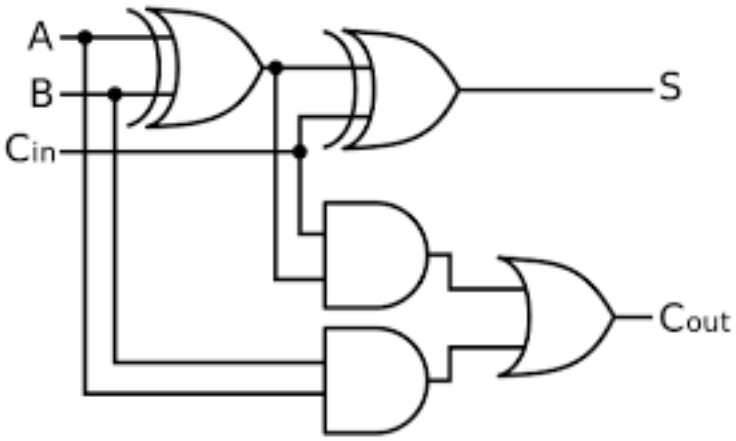
مدار نیم جمع کننده یک مدار منطقی ترکیبی است که دو بیت را با هم جمع می کند.
خروجی این مدار بیت جمع sum bit و بیت نقلی carry bit خواهد بود.

مدار تمام جمع کننده یک مدار منطقی ترکیبی است که عملیات جمع را بر روی سه بیت انجام خواهد داد.
دو بیت از این سه بیت ورودی خواهد بود و بیت سوم ما بیت نقلی ورودی carry in می باشد.

طبق شکل ۲ مدار ترکیبی Full Adder از ۲ گیت AND و ۲ گیت XOR و ۱ گیت OR تشکیل شده است.
اما ما برای ماژول بندی بهتر و منظم تر کردن کد های پروژه از این گیت ها به صورت مستقیم استفاده نکردیم و از به هم پیوستن دو مدار Half Adder در کد استفاده کرده ایم.
برای این منظور ما ابتدا ۵ بیت ورودی و خروجی مدار Full Adder را مشخص می کنیم.

سپس دو قطعه HalfAdder0 و HalfAdder1 را تعریف می کنیم.

در مرحله بعد می توانیم از port map استفاده کنیم برای به هم پیوستن خروجی های HalfAdder0 به ورودی های HalfAdder1



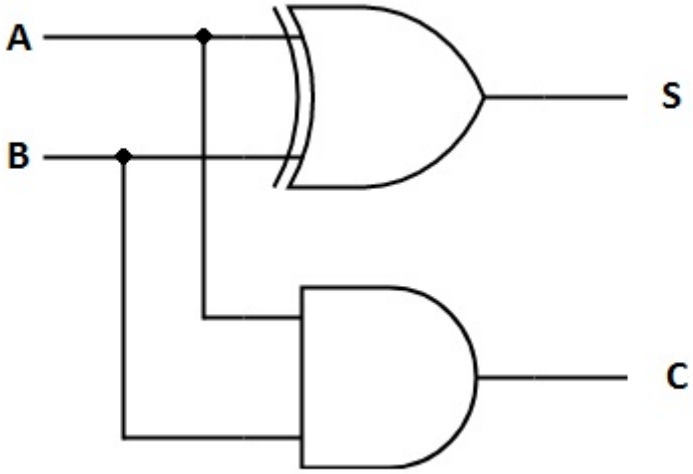
شکل ۲: مدار ترکیبی تمام جمع کننده

```
----------------------------------------------------------------------------------
-- Create Date: 11:02:53 10/18/2022
-- Design Name:
-- Module Name: FullAdder - Behavioral
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FullAdder is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           carry_in : in STD_LOGIC;
           sum : out STD_LOGIC;
           carry_out : out STD_LOGIC);
end FullAdder;

architecture Behavioral of FullAdder is
component halfadder
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           sum : out STD_LOGIC;
           carry : out STD_LOGIC);
end component;
signal S1,C1,C2 : std_logic;
begin

HALFADDER0: halfadder port map(a,b,s1,c1);
HALFADDER1: halfadder port map(carry_in,s1,sum,c2);

carry_out <= c1 or c2;

end Behavioral;
```

۳ نیم جمع کننده HalfAdder

در این آزمایش ساده ترین جز تشکیل دهنده مدار ما یک قطعه Half Adder خواهد بود.
همانطور که از آزمایش اول به خاطر داریم یک مدار Half Adder از یک گیت XOR و یک گیت AND تشکیل می شود.
کد VHDL زده شده برای این قطعه را در زیر مشاهده می کنید.
در کل برای مدار جمع کننده ۴ بیتی نیاز به ۸ قطعه Half Adder خواهیم داشت.

```
----------------------------------------------------------------------------------
-- Create Date: 11:01:20 10/18/2022
-- Design Name:
-- Module Name: HalfAdder - Behavioral
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity HalfAdder is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           sum : out STD_LOGIC;
           carry : out STD_LOGIC);
end HalfAdder;

architecture Behavioral of HalfAdder is

begin
sum <= a xor b;
carry <= a and b;

end Behavioral;
```



شکل ۳: شمای یک مدار نیم جمع کننده

۴ جمع کننده چهار بیتی FourBitAdder

در نهایت در گام آخر با به هم وصل کردن ۴ قطعه تمام جمع کننده Full Adder مدار جمع کننده ۴ بیتی ما کامل خواهد شد. همانطور که قبلا هم اشاره کرده بودیم در اینجا نیز از Portmap برای به هم وصل کردن خروجی های مدارهای اولیه به ورودی های مدارهای بعدی استفاده خواهیم کرد.

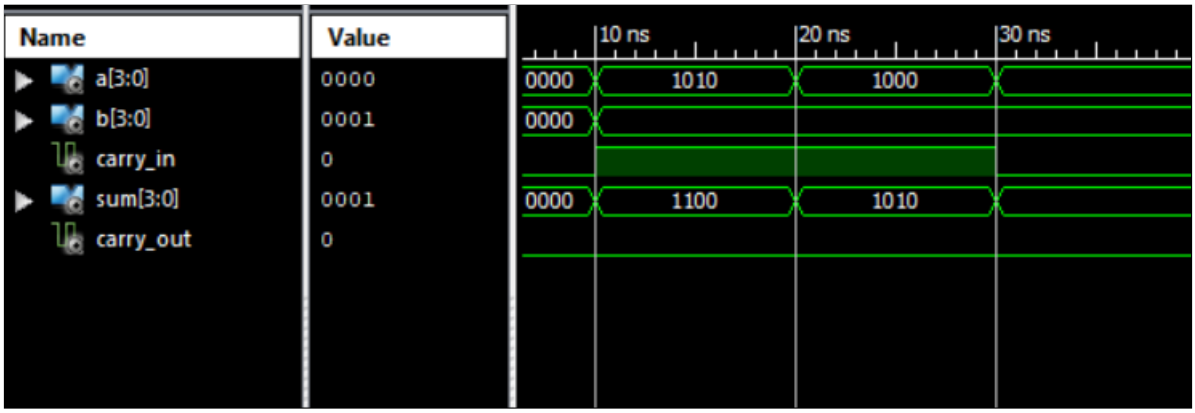
```
----------------------------------------------------------------------------------
-- Create Date: 11:20:01 10/18/2022
-- Design Name:
-- Module Name: FourBitAdder - Behavioral
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FourBitAdder is
    Port ( a : in STD_LOGIC_VECTOR (3 downto 0);
           b : in STD_LOGIC_VECTOR (3 downto 0);
           carry_in : in STD_LOGIC;
           sum : out STD_LOGIC_VECTOR (3 downto 0);
           carry_out : out STD_LOGIC);
end FourBitAdder;

architecture Behavioral of FourBitAdder is
component FullAdder
 Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           carry_in : in STD_LOGIC;
           sum : out STD_LOGIC;
           carry_out : out STD_LOGIC);
end component;

signal C : std_logic_vector(2 downto 0);


begin
fullAdder0 : FullAdder port map(a(0) , b(0) , carry_in , sum(0) , C(0));
fullAdder1 : FullAdder port map(a(1) , b(1) , C(0) , sum(1) , C(1));

fullAdder2 : FullAdder port map(a(2) , b(2) , C(1) , sum(2) , C(2));
fullAdder3 : FullAdder port map(a(3) , b(3) , C(2) , sum(3) , carry_out);


end Behavioral;
```

۵ تست کردن قطعات سخت افزاری

برای اطمینان از صحت عملکرد کد VHDL نوشته شده از Test Bench استفاده خواهیم کرد.
۳ حالت مختلف برای تست را در کد زیر در نظر گرفته ایم.
شکل موج این ۳ حالت را نیز در شکل ۴ نمایش داده ایم.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY FourAdderTestBench IS
END FourAdderTestBench;

ARCHITECTURE behavior OF FourAdderTestBench IS
    COMPONENT FourBitAdder
    PORT(
         a : IN std_logic_vector(3 downto 0);
         b : IN std_logic_vector(3 downto 0);
         carry_in : IN std_logic;
         sum : OUT std_logic_vector(3 downto 0);
         carry_out : OUT std_logic
        );
    END COMPONENT;

   signal a : std_logic_vector(3 downto 0) := (others => '0');
   signal b : std_logic_vector(3 downto 0) := (others => '0');
   signal carry_in : std_logic := '0';

   signal sum : std_logic_vector(3 downto 0);
   signal carry_out : std_logic;
```

```
BEGIN
   uut: FourBitAdder PORT MAP (
         a => a,
         b => b,
         carry_in => carry_in,
         sum => sum,
         carry_out => carry_out
       );

   stim_proc: process
   begin
      wait for 10 ns;
      a <= "1010";
      b <= "0001";

      carry_in <= '1';

      -- Test(2)
       wait for 10 ns;
      a <= "1000";
      b <= "0001";

      carry_in <= '1';

      -- Test(3)
       wait for 10 ns;
      a <= "0000";
      b <= "0001";

      carry_in <= '0';

      wait;
   end process;

END;
```



شکل ۴: شکل موج حاصل از جمع چند عدد