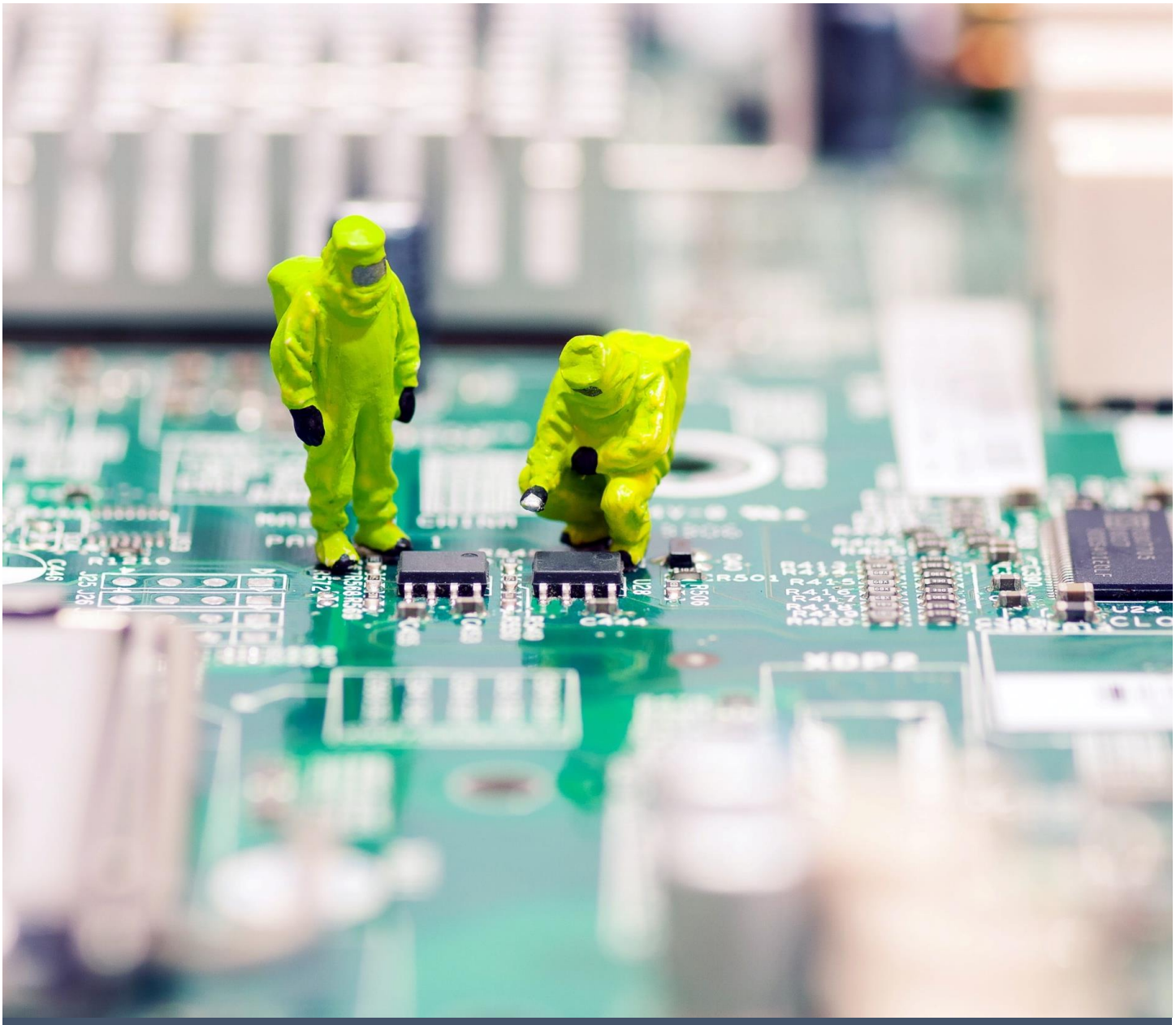


MH

# Masoud Heidary

think outside the box

## CA Lab – EX3

Masoud-Heidary.ir MasoudHeidaryMH@gmail.com MasoudHeidaryMH@outlook.com

طراحی را از پایین به بالا، به صورت زیر شروع میکنیم

ابتدا ماژول جمع کننده تک بیتی را به صورت زیر طراحی میکنیم

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

entity FA is
    Port ( A : in        STD_LOGIC;
           B : in        STD_LOGIC;
           Cin : in      STD_LOGIC;
           Sum : out     STD_LOGIC;
           Cout : out    STD_LOGIC);
end FA;

architecture Behavioral of FA is
begin
    Sum <= A xor B xor Cin;
   Cout <= (A and B) or (B and Cin) or (A and Cin);

end Behavioral;
```

سپس ماژول جمع کننده 8 بیتی را با استفاده از آن به صورت زیر میسازیم

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

entity ADD8 is
    Port ( A : in        STD_LOGIC_VECTOR (7 downto 0);
           B : in        STD_LOGIC_VECTOR (7 downto 0);
           Cin : in      STD_LOGIC;
           Sum : out     STD_LOGIC_VECTOR (7 downto 0);
           Cout : out    STD_LOGIC);
end ADD8;

architecture Behavioral of ADD8 is
    signal c: STD_LOGIC_VECTOR (6 downto 0);

    COMPONENT FA
    PORT(
```

```
        A : IN      std_logic;
        B : IN      std_logic;
        Cin : IN    std_logic;
        Sum : OUT   std_logic;
        Cout : OUT  std_logic
        );
    END COMPONENT;
begin

-- FA 0
Inst_FA0: FA PORT MAP (
        A => A(0),
        B => B(0),
        Cin => Cin,
        Sum => Sum(0),
        Cout => c(0)
    );

-- FA from 1 to 6
gen : for i in 1 to 6 generate
    Inst_FA: FA PORT MAP (
        A => A(i),
        B => B(i),
        Cin => c(i-1),
        Sum => Sum(i),
        Cout => c(i)
    );
end generate ; -- gen

-- FA 7
Inst_FA7: FA PORT MAP (
        A => A(7),
        B => B(7),
        Cin => c(6),
        Sum => Sum(7),
        Cout => Cout
    );

end Behavioral;
```

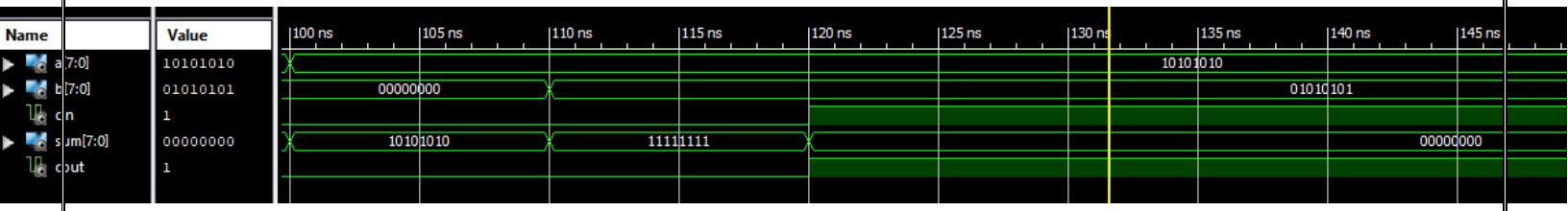
همان طور که واضح است، برای راحتی کار، از generate استفاده کرده ایم.

برای تست جمع کننده ای که ساخته ایم، به صورت زیر عمل میکنیم

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY ADD8_test IS
END ADD8_test;

ARCHITECTURE behavior OF ADD8_test IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT ADD8
    PORT(
         A : IN  std_logic_vector(7 downto 0);
         B : IN  std_logic_vector(7 downto 0);
         Cin : IN  std_logic;
         Sum : OUT  std_logic_vector(7 downto 0);
         Cout : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal A : std_logic_vector(7 downto 0) := (others => '0');
   signal B : std_logic_vector(7 downto 0) := (others => '0');
   signal Cin : std_logic := '0';

   --Outputs
   signal Sum : std_logic_vector(7 downto 0);
   signal Cout : std_logic;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: ADD8 PORT MAP (
          A => A,
          B => B,
          Cin => Cin,
          Sum => Sum,
          Cout => Cout
```

```
        );


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      A <= "10101010";
      B <= "00000000";
      Cin <= '0';
      wait for 10 ns;

      B <= "01010101";
      wait for 10 ns;

      Cin <= '1';
      wait for 10 ns;
      wait;
   end process;

END;
```

خروجی ما به صورت زیر میشود




$$10101010 + 00000000 \rightarrow 10101010\ (ok)$$

$$10101010 + 01010101 \rightarrow 11111111\ (ok)$$

$$+ 1\ (carry) \rightarrow 1\_0000\_0000\ (ok)$$

طراحی ALU

طراحی را به صورت این پیاده سازی میکنیم، که با استفاده از select, when سیگنال مناسب را به خروجی ارسال میکنیم، و برای محاسبات میانی نیز از سیگنال های میانی استفاده میکنیم

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


entity ALU is
    Port ( A : in       STD_LOGIC_VECTOR (7 downto 0);
           B : in       STD_LOGIC_VECTOR (7 downto 0);
           F : out      STD_LOGIC_VECTOR (7 downto 0);
           Cin : in     STD_LOGIC;
           OPCODE : in  STD_LOGIC_VECTOR (3 downto 0);
           Cout : out   STD_LOGIC);
end ALU;

architecture Behavioral of ALU is

    -- signals
    signal A_add_Cin:   STD_LOGIC_VECTOR (7 downto 0);
    signal A_add_B:     STD_LOGIC_VECTOR (7 downto 0);
    signal Sub:         STD_LOGIC_VECTOR (7 downto 0);
    signal R_SHIFT:     STD_LOGIC_VECTOR (7 downto 0);
    signal L_SHIFT:     STD_LOGIC_VECTOR (7 downto 0);

    signal Carry:       STD_LOGIC_VECTOR (2 downto 0);

    COMPONENT ADD8
    PORT(
        A : IN      std_logic_vector(7 downto 0);
        B : IN      std_logic_vector(7 downto 0);
        Cin : IN    std_logic;
        Sum : OUT   std_logic_vector(7 downto 0);
        Cout : OUT  std_logic
        );
    END COMPONENT;
begin
```

```
    -- A + Cin
    Inst_ADD8_0: ADD8 PORT MAP(
        A => A,
        B => "00000000",
        Cin => Cin,
        Sum => A_add_Cin,
        Cout => Carry(0)
    );

    -- A + B + Cin
    Inst_ADD8_1: ADD8 PORT MAP(
        A => A,
        B => B,
        Cin => Cin,
        Sum => A_add_B,
        Cout => Carry(1)
    );

    -- A + (not B) + Cin
    Inst_ADD8_2: ADD8 PORT MAP(
        A => A,
        B => (not B),
        Cin => Cin,
        Sum => Sub,
        Cout => Carry(2)
    );

    -- shift to right
    R: for i in 0 to 6 generate
        R_SHIFT(i) <= A(i+1);
    end generate;
        R_SHIFT(7) <= '0';

    -- shift to left
    L: for i in 1 to 7 generate
        L_SHIFT(i) <= A(i-1);
    end generate;
        L_SHIFT(0) <= '0';


    with OPCODE (3 downto 0) select
        F <=    A_add_Cin   when "0000",
                A_add_B     when "0001",
                Sub         when "0010",
                A-1       when "0011",
```

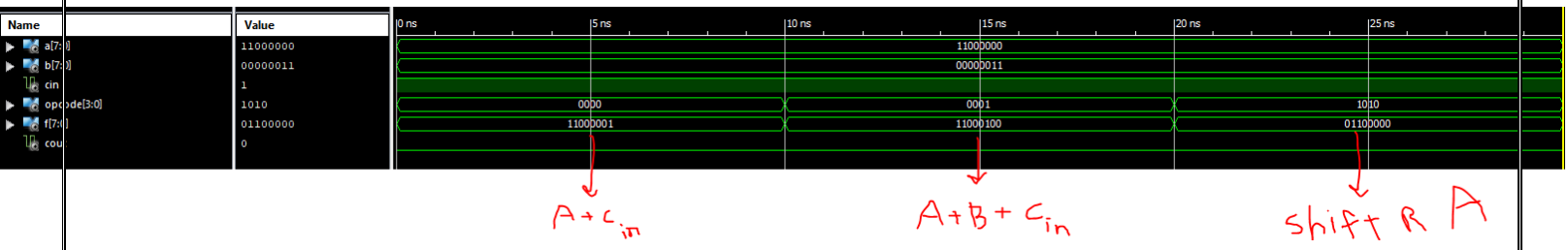
```
                A and B     when "0100",
                A or B      when "0101",
                A xor B     when "0110",
                not A       when "0111",
                R_SHIFT     when "1000",
                R_SHIFT     when "1001",
                R_SHIFT     when "1010",
                R_SHIFT     when "1011",
                L_SHIFT     when "1100",
                L_SHIFT     when "1101",
                L_SHIFT     when "1110",
                L_SHIFT     when "1111",
                "ZZZZZZZZ"  when others;

    with OPCODE (3 downto 0) select
        Cout <= Carry(0) when "0000",
                Carry(1) when "0001",
                Carry(2) when "0010",
                '0'      when others;

end Behavioral;
```

برای تست کد نوشته شده، به صورت بسیار پایه ای، به صورت زیر عمل میکنیم

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY ALU_test IS
END ALU_test;

ARCHITECTURE behavior OF ALU_test IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT ALU
    PORT(
         A : IN  std_logic_vector(7 downto 0);
         B : IN  std_logic_vector(7 downto 0);
         F : OUT  std_logic_vector(7 downto 0);
         Cin : IN  std_logic;
         OPCODE : IN  std_logic_vector(3 downto 0);
         Cout : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal A : std_logic_vector(7 downto 0) := (others => '0');
   signal B : std_logic_vector(7 downto 0) := (others => '0');
   signal Cin : std_logic := '0';
   signal OPCODE : std_logic_vector(3 downto 0) := (others => '0');

   --Outputs
   signal F : std_logic_vector(7 downto 0);
   signal Cout : std_logic;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: ALU PORT MAP (
          A => A,
          B => B,
          F => F,
          Cin => Cin,
          OPCODE => OPCODE,
          Cout => Cout
```

```
        );

   -- Stimulus process
   stim_proc: process
   begin
      A <= "11000000";
      B <= "00000011";
      Cin <= '1';

      OPCODE <= "0000";
      wait for 10 ns;

      OPCODE <= "0001";
      wait for 10 ns;

      OPCODE <= "1010";
      wait for 10 ns;

      wait;
   end process;

END;
```

خروجی تست ما به صورت زیر میشود




همان طور که واضح است، سه تابع تست شده به درستی کار میکنند.

در واقع باید تمام توابع این ماژول را با ورودی های مختلف تست کنیم، و چون اینکار به صورت دستی و با نمودار ممکن نمیباشد، باید از framework ها استفاده کنیم که از آن صرف نظر میکنیم.