北京理工大學
Beijing Institute of Technology

本科实验报告

实验名称：十六位加法器设计

课程名称：	集成电路设计实践（Ⅰ）	实验时间：	2018/4/22
任课教师：	张蕾	实验地点：	综 A
实验教师：	张蕾	实验类型：	■ 原理验证 □ 综合设计 □ 自主创新
学生姓名：	施念		
学号/班级：	1120161302/05011609	组　　号：	
学　　院：	信息与电子学院	同组搭档：	
专　　业：	电子信息工程	成　　绩：	

实验二 十六位加法器设计

一、实验目的

（1）掌握元件例化的方法

（2）理解 for/generate 语句的用法

（3）编程完成 4 位加法器和 16 位加法器的设计

二、实验原理

将 n 个加法器相连可得 n 位加法器，但是加法时间较长，解决的方法之一就是采用“超前进位加法器电路”来同时形成各位进位，从而实现快速加法。超前进位产生电路是根据各位进位的形成条件来实现的。

以一个四位超前进位加法器为例，首先需要做出两个传递函数：进位产生函数 Pi 和进位传送函数 Qi。

$$P_i = A_i \oplus B_i$$

$$Q_i = A_i \,\&\, B_i$$

如下图所示，为 4 位超前进位加法器内部逻辑图以及各个门所代表的含义，由下图可以写出各个进位输入的逻辑表达式

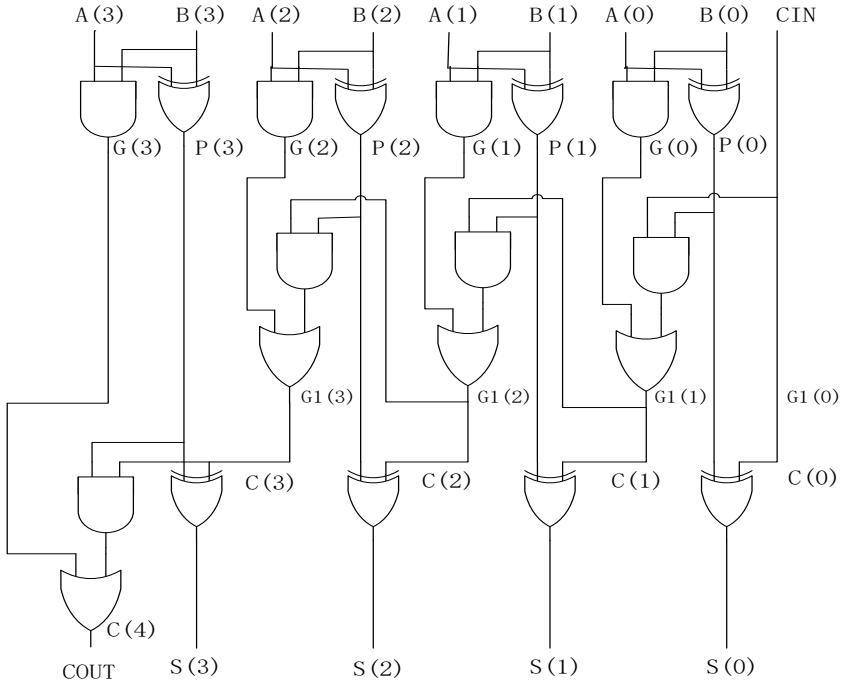


图 1 四位超前进位加法器内部逻辑图

图 2 逻辑门示意图

由上图可以写出各个进位输入的逻辑表达式：

c(0) <= cin;

c(1) <= (cin and p(0)) or g(0) ;

c(2) <= (cin and p(0) and p(1)) or (g(0) and p(1)) or g(1);

c(3) <= (cin and p(0) and p(1) and P(2)) or (g(0) and p(1) and p(2)) or (g(1) and p(2)) or g(2);

c(4) <= (cin and p(0) and p(1) and p(2) and p(3)) or (g(0) and p(1) and p(2) and p(3)) or (g(1) and p(2) and p(3)) or (g(2) and p(3)) or g(3);

而每位的输入可以写成表达式：

$$S_i = P_i \oplus C_i$$

二、实验内容

本次任务需要使用 VHDL 语言编写一个 16 位超前进位加法器，顶层模块的名字为 adder.vhd，建议调用四个子模块，每个子模块进行 4 位超前进位加法，最终整个设计完成 16 位加法。进行仿真时需要至少 4 个测试值，包含 cin 为高和低的情况。

16 位加法器的设计：

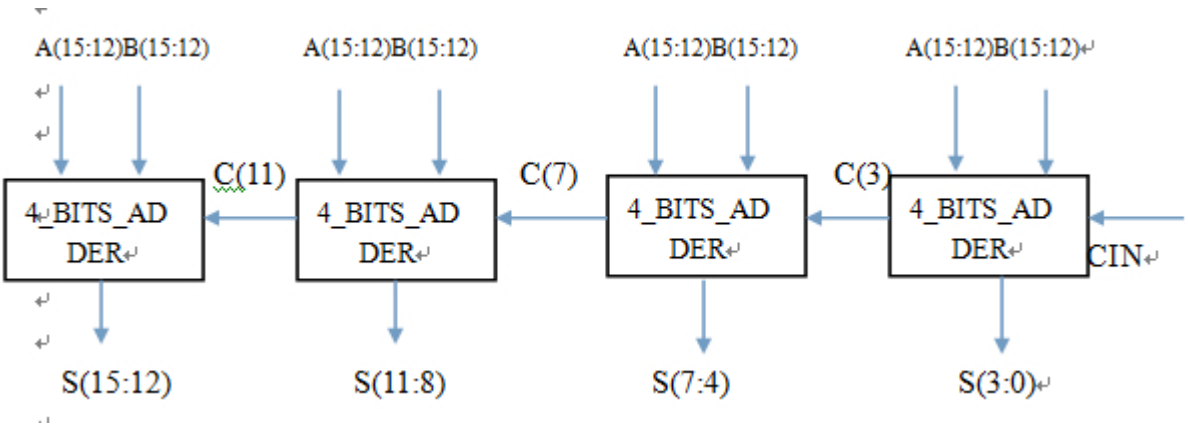


图 3 16 位加法器结构示意图

三、实验代码

----------adder_4.vhd----------

```
library ieee;
use ieee.std_logic_1164.all;
------------------------------------------------------------
entity adder_4 is
port (
      a,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 entity adder_4;
------------------------------------------------------------
architecture behave of adder_4 is
signal c: std_logic_vector(4 downto 0);
signal p: std_logic_vector(3 downto 0) ;
signal g: std_logic_vector(3 downto 0) ;
begin
------------------------------------------------------------
G1 : for i in 3 downto 0 generate
      p(i) <= a(i) xor b(i);
      g(i) <= a(i) and b(i);
      s(i) <= p(i) xor c(i);
end generate G1;
------------------------------------------------------------
c(0) <= cin; --initialize c(0)
G2 : for j in 1 to 4 generate
      c(j) <= (c(j-1) and p(j-1)) or g(j-1);
end generate G2;
cout <= c(4); -- cout
end architecture behave;
```

----------adder.vhd----------

```
library ieee;
use ieee.std_logic_1164.all;
------------------------------------------------------------
entity adder is
      port (
            a,b:in std_logic_vector(15 downto 0) ;
```

```
            s:out std_logic_vector(15 downto 0) ;
            cin: in std_logic;
            cout:out std_logic
      );
end entity adder;
----------------------------------------------------------------
architecture behave of adder is
component adder_4 is
      port (
      a,b:in std_logic_vector(3 downto 0);
      s:out std_logic_vector(3 downto 0);
      cin:in std_logic;
      cout: out std_logic
      );
end component adder_4;
----------------------------------------------------------------
signal m1, m2, m3:std_logic;
begin
u1:adder_4 port map(a(3 downto 0),b(3 downto 0),s(3 downto 0),cin,m1);
u2:adder_4 port map(a(7 downto 4),b(7 downto 4),s(7 downto 4),m1,m2);
u3:adder_4 port map(a(11 downto 8),b(11 downto 8),s(11 downto 8),m2,m3);
u4:adder_4 port map(a(15 downto 12),b(15 downto 12),s(15 downto 12),m3,cout);
end architecture behave;
```

----------adder_tb.vhd----------

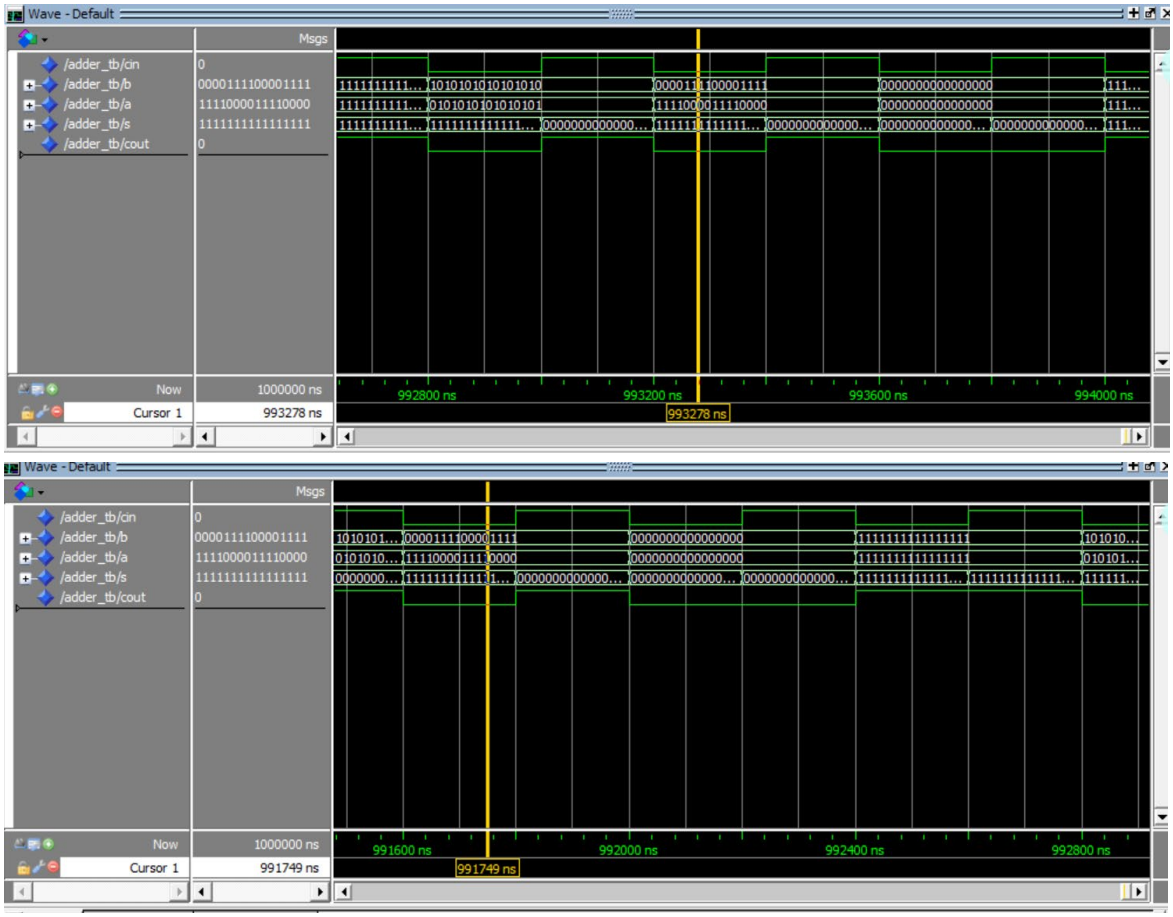
```
library ieee;
use ieee.std_logic_1164.all;
----------------------------------------
entity adder_tb is
end adder_tb;
----------------------------------------
architecture adder_tb of adder_tb is
component adder
   port (a,b:in std_logic_vector(15 downto 0);
            cin:in std_logic;
            s:out std_logic_vector(15 downto 0);
            cout: out std_logic);
end component;
----------------------------------------

signal a,b: std_logic_vector(15 downto 0);
```

```
signal cin:std_logic;
signal s:std_logic_vector(15 downto 0);
signal cout:std_logic;

begin
dut:adder port map
(a(15 downto 0)=>a(15 downto 0),b(15 downto 0)=>b(15 downto 0),cin=>cin,s(15 downto
0)=>s(15 downto 0),cout=>cout);
-----------------------------------------
a_gen:process
begin
   a<="0000000000000000";
   wait for    400 ns;
      a<="1111111111111111";
      wait for    400 ns;
            a<="0101010101010101";
         wait for    400 ns;
                  a<="1111000011110000";
            wait for    400 ns;
end process;
-----------------------------------------
b_gen:process
begin
   b<="0000000000000000";
   wait for    400 ns;
      b<="1111111111111111";
      wait for    400 ns;
            b<="1010101010101010";
         wait for    400 ns;
                  b<="0000111100001111";
            wait for    400 ns;
end process;
-----------------------------------------
cin_gen:process
begin
   cin<='0';
   wait for 200 ns;
   cin<='1';
   wait for 200 ns;
end process;
end adder_tb;
-------------------------------------------------
```

四、仿真结果



五、分析与总结

从结果我们可以看出，当 cin 为 1 的时候，若不发生最高位进位（即 cout≠1），则此时的运算结果（可以转化为十进制数据类型）是正确的，当不发生最高位进位时，结果与预期一致。

总的来说，利用 4 位超前进位加法实现了 16 位超前进位加法的功能。