实验二 多路复用器与加法器的实现

一、实验目的

1．熟悉多路复用器、加法器的工作原理。

2．学会使用 VHDL 语言设计多路复用器、加法器。

3．掌握 generic 的使用，设计 n-1 多路复用器。

4．兼顾速度与成本，设计行波加法器和先行进位加法器。

二、实验背景

1．多路复用器，又名多路选择器、多路开关

多路复用器是一个组合电路，它可以从多个输入中选择一个输入，并将信息直接传输到输出。选择哪一条输入线由一组输入变量控制，它们被称为选择输入。通常，2^n 条输入线要 n 个选择输入，选择输入的位组合决定选择哪个输入线。例如 n=1 的 2-1 多路复用器。这个复用器有两个信息输入 I0 和 I1，一个单独的选择输入 S，电路的真值表如表 1 所示。

表 1　2-1 多路复用器真值表

S	I_0	I_1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

分析真值表可知，如果选择输入 S=0，多路复用器输出为 I0 的值；如果选择输入 S=1，多路开关输出 I1 的值。这样，S 不是选择输入 I0 就是选择输入 I1 到输出 Y。通过这些讨论，可以看出，2-1 多路复用器输出 Y 的方程式为

$$Y = \overline{S}I_0 + SI_1$$

2．运用 generic 设计 n 位加法器，再调用 n 位加法器实现 4 位加法器。

a）n 位加法器

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity par_add is
  generic(n:integer);
  port(a,b:in std_logic_vector(n-1 downto 0);
       s:out std_logic_vector(n-1 downto 0);
       c:out std_logic);
end par_add;

architecture exa of par_add is
signal t:std_logic_vector(n downto 0);
begin
   t<=('0'&a)+('0'&b);
   s<=t(n-1 downto 0);
   c<=t(n);
end exa;
```

b）调用 n 位加法器，定制 4 位加法器

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity add3 is
  port(x,y:in std_logic_vector(3 downto 0);
       cin:in std_logic;
       s:out std_logic_vector(3 downto 0);
       cout:out std_logic);
end add3;

architecture exa of add3 is
component par_add is
  generic(n:integer);
  port(a,b:in std_logic_vector(n-1 downto 0);
       s:out std_logic_vector(n-1 downto 0);
       c:out std_logic);
end component;
signal sum:std_logic_vector(3 downto 0);
signal mid:std_logic_vector(4 downto 0);
signal c3:std_logic;
begin
    g0:par_add generic map(n=>4)port map(a=>x,b=>y,s=>sum,c=>c3);
    mid<=(c3&sum)+("0000"&cin);
```

```
        s<=mid(3 downto 0);
        cout<=mid(4);
    end exa;
```

3．行波进位加法器

全加器

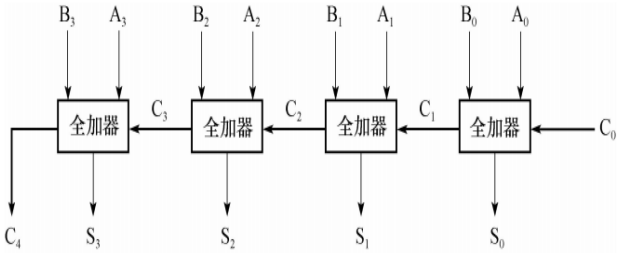
全加器是实现三位数相加的组合逻辑电路，共有三个输入，两个输出。输入变量中的两个用 X 和 Y 表示，代表两个加数，第三个输入 Z 表示低位产生的进位。两个输出用 S(和)与 C(进位)来表示。输出值由三位输入的算术和决定。当所有输入都为 0 时，输出均为 0。当输入仅有一个为 1 或三个全为 1 时，输出 S 为 1。当输入有两个或三个为 1 时，则输出 C 为 1。全加器的布尔表达式用异或运算表示可写成：

$S=X⊕Y⊕Z$

$C=XY+YZ+XZ$

四位行波进位加法器

四位行波进位加法器由四个全加器级联形成。被加数A和加数B的下标从右至左依次递增，下标0表示最低有效位，进位位将整个全加器链式地连接起来。并行加法器的进位输入为C_0，进位输出为C_4。一个n位的行波进位加法器需要n个全加器，每个进位输出连接到下一个高位全加器的进位输入。



4、先行进位加法器

行波进位加法器需要一级一级的进位，进位延迟很大。先行进位加法器（也叫超前进位加法器）可以有效的减少进位延迟。

设二进制加法器的第 i 位输入为 X_i、Y_i，输出为 S_i，进位输入为 C_i，进位输出为 C_{i+1}

则有

$$Si = Xi⊕Yi⊕Ci$$

$$Ci+1 = Xi \cdot Yi + Xi \cdot Ci + Yi \cdot Ci = Xi \cdot Yi + (Xi + Yi) \cdot Ci$$

令 Gi = Xi • Yi,　Pi = Xi+Yi

则 Ci+1 = Gi + Pi • Ci

当 Xi 和 Yi 都为 1 时，Gi = 1，产生进位 $C_{i+1} = 1$

当 Xi 和 Yi 有一个为 1 时，Pi = 1，传递进位 $C_{i+1} = Ci$

因此 Gi 定义为进位产生信号，Pi 定义为进位传递信号。Gi 的优先级比 Pi 高，也就是说：当 Gi = 1 时（当然此时也有 Pi = 1 ），无条件产生进位，而不管 Ci 是多少;

当 Gi=0 而 Pi=1 时，进位输出为 Ci, 跟 Ci 之前的逻辑有关。

下面推导 4 位超前进位加法器。设 4 位加数和被加数为 A 和 B，进位输入为 Cin, 进位输出为 Cout, 对于第 i 位的进位产生 Gi = Ai • Bi , 进位传递 Pi=Ai+Bi , i=0,1,2,3

这各级进位输出，递归的展开 Ci ，有：

C0 = Cin

C1=G0 + P0 • C0

C2=G1 + P1 • C1 = G1 + P1 •（G0 + P0 • C0）=G1 + P1 • G0 + P1 • P0 •C0

C3=G2 + P2 • C2 = G2 + P2 • G1 + P2 • P1 • G0 + P2 • P1 • P0 • C0

C4=G3 + P3 • C3 = G3 + P3 • G2 + P3 • P2 • G1 + P3 • P2 • P1 • G0 + P3 • P2 • P1 • P0 • C0

Cout=C4

由此可以看出，各级的进位彼此独立产生，只与输入数据和 Cin 有关，将各级间的进位级联传播给去掉了，因此减小了进位产生的延迟。

每个等式与只有三级延迟的电路对应，第一级延迟对应进位产生信号和进位传递信号，后两级延迟对应上面的积之和，这可从 RTL 视图中看到。

同时由真值表可以简单地得出第 i 位的和为：

$$Si = Xi \oplus Yi \oplus Ci = (Xi \cdot Yi) \oplus (Xi+Yi) \oplus Ci = Gi \oplus Pi \oplus Ci$$

三、实验内容

1．用 VHDL 语言设计 8 重 3-1 多路复用器；

2．用 VHDL 语言设计 n-1 多路复用器，调用该 n-1 多路复用器定制为 8-1 多路复用器。

3．用 VHDL 语言设计 4 位行波进位加法器。

4．用 VHDL 语言设计 4 位先行进位加法器。

四、实验要求

1．进实验室前，请写一份预习报告；如有疑问，可在学习通平台相互讨论。

2．预习报告内容有:

- 8 重 3-1 多路复用器的 VHDL 程序；
- n-1 多路复用器，调用该 n-1 多路复用器定制为 8-1 多路复用器的 VHDL 程序；
- 4 位行波进位加法器的 VHDL 程序。
- 4 位先行进位加法器的 VHDL 程序。

3．在文本编辑区使用 VHDL 硬件描述语言设计功能块，再利用波形编辑区进行仿真，以此验证电路的逻辑功能是否正确，最后在 Tool 下用 netlist viewer 查看 RTL viewer，以查看实现的 RTL 电路图。

5．实验结束前，由指导老师检查了仿真波形后方可离开。

6、最后撰写实验报告，提交到学习通平台，并在平台上分享设计的警告、资源消耗以及 RTL 视图。

7、评判各种实现方案，并打分。

五、思考题

1．多路复用器的实现方法很多，请总结两种以上实现方法。

2．总结 VHDL 语言描述多路复用器的方法和常用语句。