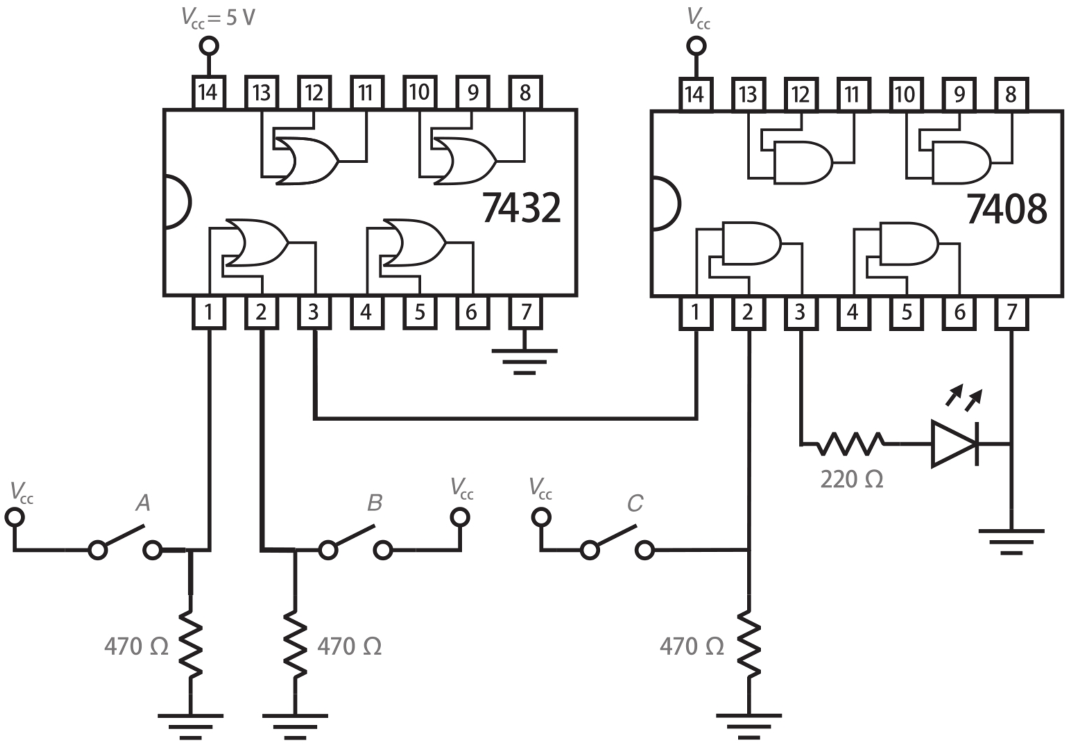


图4-23 （*A* OR *B*）AND *C*的连线图，添加开关以控制输入

如果是在面包板上搭建的话，完整的电路将类似于图4-24所示。在我的电路中，我用按钮作开关。如果碰巧你仔细观察了，你可能会看到照片中的下拉电阻的阻值是1kΩ，与图4-23中建议的470Ω不同，但两者都可以工作。

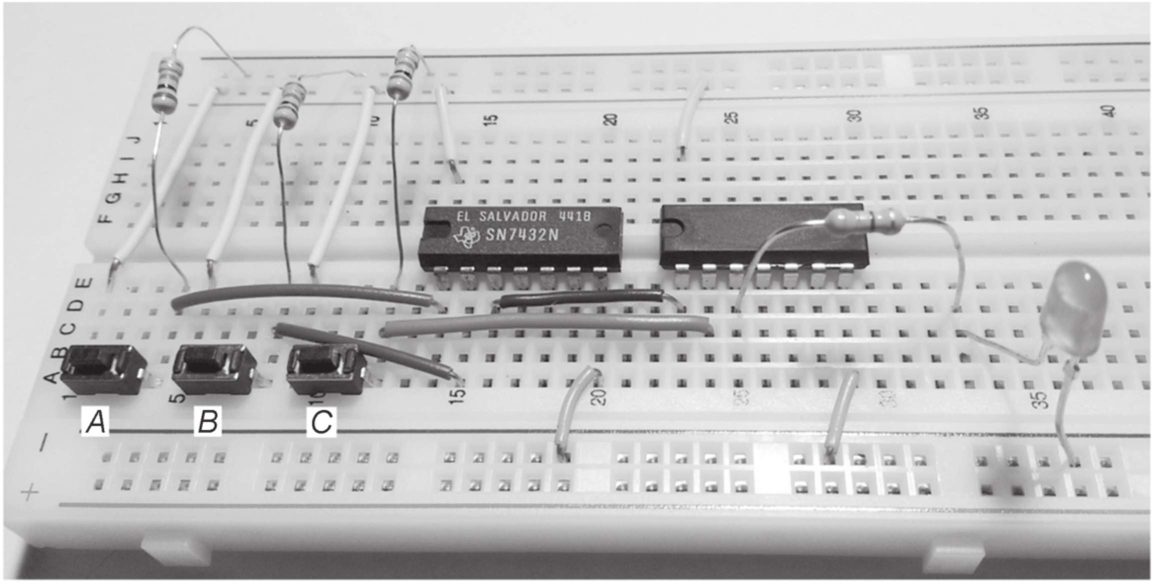


图4-24 在面包板上实现（A OR B）AND C

按图4-24搭建好电路后，请务必检查输入的各种组合，看看是否匹配表4-4所给的（*A* OR *B*）AND *C*真值表。如果电路没有按预期的工作，请查阅附录B部分排除故障。

第5章

数字电路中的算术运算

在第4章中，我们介绍了数字电路和逻辑门，它们能让我们用硬件实现逻辑表达式。在本书的前面，我们把计算机定义为可以通过编程来执行一组指令的电子设备。在本章中，我将通过展示简单的逻辑门是如何为计算机执行的运算铺平道路，把这些概念连接起来。我们将讨论所有计算机都会执行的一种运算——加法运算。首先，我们复习二进制加法的基础知识。然后，我们用逻辑门搭建加法运算硬件，演示计算机中简单的门如何协同工作来执行有用的操作。最后，我们将讨论计算机中有符号和无符号整数的表示。

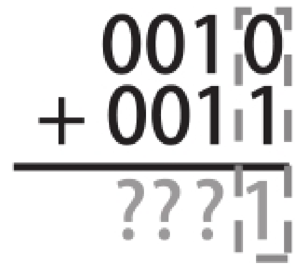
5.1 二进制加法

让我们来看看二进制中加法的基础知识。加法的基本原理在所有的位值系统中都是一样的，所以你已经有了一个良好的开端，因为你已经知道如何在十进制中进行加法运算！与其讨论抽象概念，不如举个具体的例子：将两个二进制4位数0010和0011相加，如图5-1所示。

和在十进制中一样，我们从最右边的位置（称为最低有效位）开始，把两个值相加（见图5-2），这里是0+1=1。

$$
\begin{array}{r}
0010 \\
+\,0011 \\
\hline
????
\end{array}
$$

▲图5-1 两个二进制数相加

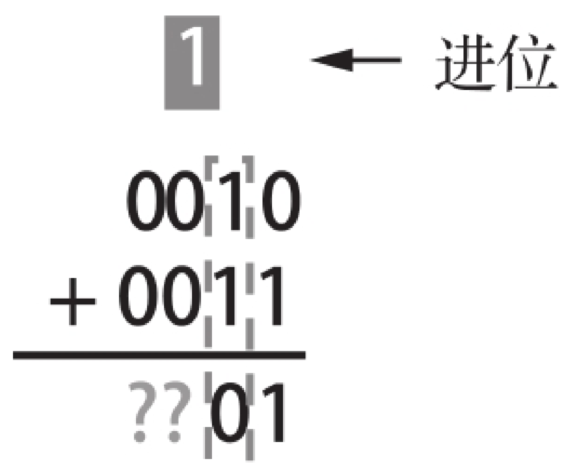


▲图5-2 两个二进制数的最低有效位相加

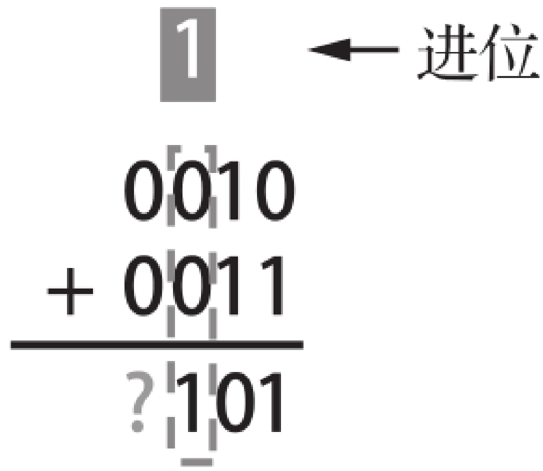
现在向左移动一位，再把这些值相加，如图5-3所示。

如图5-3所示，这个位置需要计算1+1，这给我们带来一个有趣的转折。在十进制中，我们用符号2表示1+1，但在二进制中我们只有两个符号：0和1。在二进制中，1+1等于10（解释参见第1章），需要用两个位来表示，但一个位置上只能有1位，所以将0放在当前位置上，将1进位到下一个位置，如图5-3所示。现在，我们可以移到下一个位置（见图5-4），当我们相加这些位时，必须也把前一个位置的进位加进来，由此得到1+0+0=1。

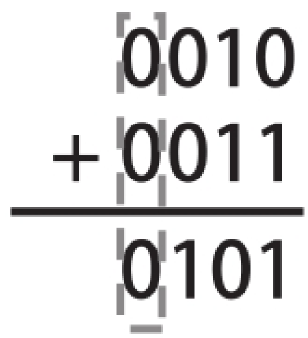
最后，我们将最高有效位相加，如图5-5所示。



▲图5-3 2的位置相加



▲图5-4 4的位置相加



▲图5-5 8的位置相加

当所有位都加完后，便得到了完整的二进制结果0101。检查结果正确与否的一种方法是全部转换成十进制，如图5-6所示。

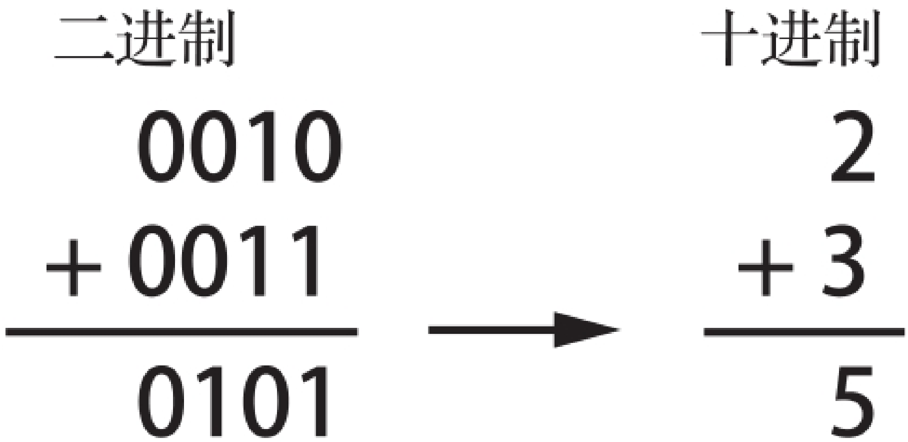


图5-6 两个二进制数相加，然后将各数及结果全部转换成十进制

如图5-6所示，二进制答案（0101）和预期的十进制答案（5）一致。很简单！

练习5-1：练习二进制加法

你现在可以练习一下刚刚学到的知识。尝试进行下列加法运算：

0001+0010=____

0011+0001=____

0101+0011=____

0111+0011=____

幸运的是，不管基数是什么，加法的运算方式都是一样的。基数之间唯一的不同是有多少符号可以使用。二进制使得加法特别简单，因为每个位置上的加法运算总是恰好产生两个输出位，每个位都只有两种可能的值：

❑输出1 为0或1的和数位（*S*），表示加法运算结果的最低有效位。

❑输出2 为0或1的进位位（*C*）。

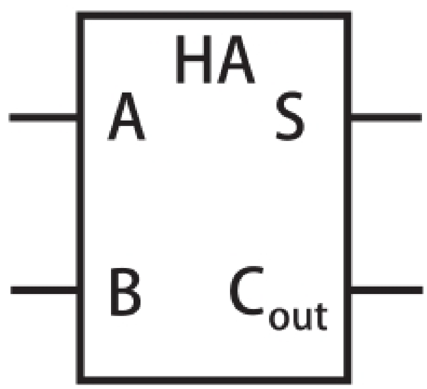
out

5.2 半加器

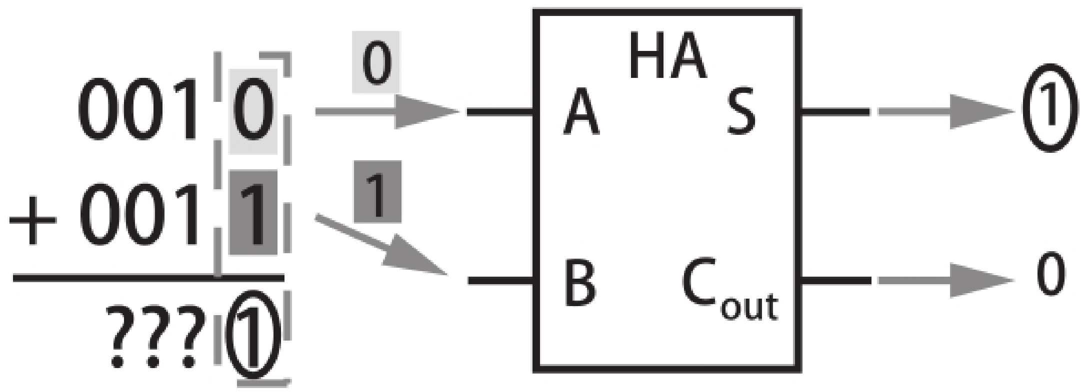
假设我们想要构建一个数字电路，把两个二进制数的某个位置加起来。开始时，我们关注最低有效位。将两个数的最低有效位相加只需要两个二进制输入（称为*A*和*B*），其二进制输出是一个和数位（*S*）和一个进位位（*C*）。我们把这样的电路称为半加器。图5-7展示了半加器符号。

out

为了说清楚半加器是如何与我们之前的两个二进制数相加示例相适应的，图5-8把这两个概念联系了起来。



▲图5-7 半加器符号



▲图5-8 半加器的运作

如图5-8所示，第一个数的最低有效位是输入*A*，第二个数的最低有效位是输入*B*。和数是一个输出*S*，进位也是一个输出。

半加器的内部可以实现为一个组合逻辑电路，所以我们可以用真值表描述它，如表5-1所示。注意，*A*和*B*是输入，而*S*和*C*是输出。

out

表5-1 半加器真值表

输入		输出	
A	*B*	*S*	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

让我们来看看表5-1中的真值表。0加0得0，无进位。0加1（或1加0）得1，无进位。1加1得0，进位为1。

现在，怎样用数字逻辑门来实现它呢？如果单看一个输出，那么解决方案很简单，如图5-9所示。

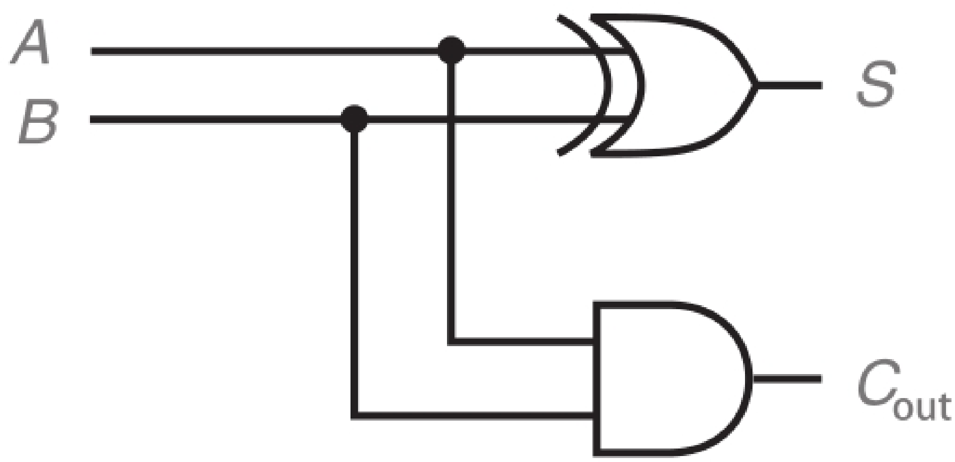
只看图5-9的输出*S*的话，可以发现它与XOR门的真值表（参见第4章）完全匹配。只看*C*的话，可以发现它与AND门的输出匹配。因此，仅用XOR和AND两个门就可以实现半加器，如图5-10所示。

out

输入		输出	
A	*B*	*S*	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

XOR　　AND

▲图5-9 半加器真值表的输出分别匹配XOR和AND



▲图5-10 用XOR和AND两个逻辑门实现的半加器

如图5-10所示，数字输入*A*和*B*充当XOR门和AND门的输入。这两个门产生所需的输出*S*和*C*。

out

注意

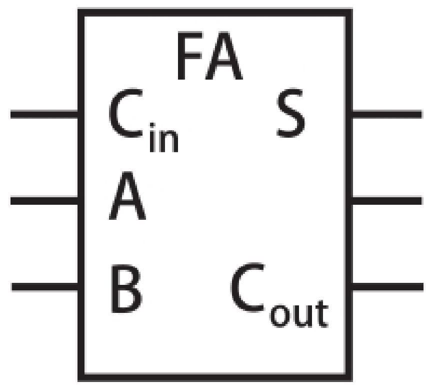
参阅设计5搭建半加器电路。

5.3 全加器

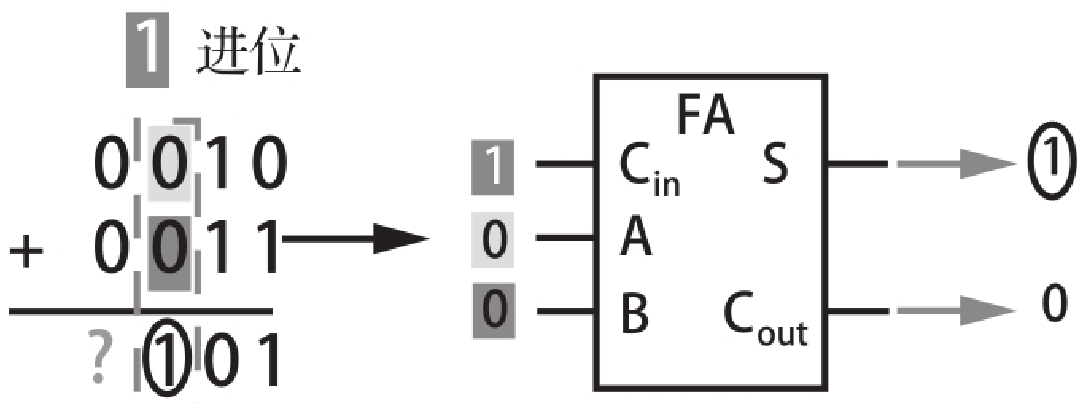
半加器可以处理两个二进制数的最低有效位加法运算。但是，每个后续位都需要一个额外的输入：进位位*C*。这是因为，除了最低有效位之外，每个位都需要处理一种情况，即前一个位的加法运算产生了进位，这个进位反过来又成为当前位的进位。增加*C*输入需要新的电路设计，我们把这种电路称为全加器。如图5-11所示，全加器的符号类似于半加器的符号，不同之处仅在于多了一个额外的输入*C*。

ininin

图5-12给出了一个单个位二进制加法与全加器关系的例子。



▲图5-11 全加器符号



▲图5-12 全加器的运作

全加器处理包含进位位的单个位加法。在图5-12所示的例子中，我们进行4的位置上的加法运算。由于前一个位置上是1和1，因此产生进位1。在当前位置，全加器接收3个输入（*A*=0，*B*=0，*C*=1），产生输出*S*=1和*C*=0。

inout

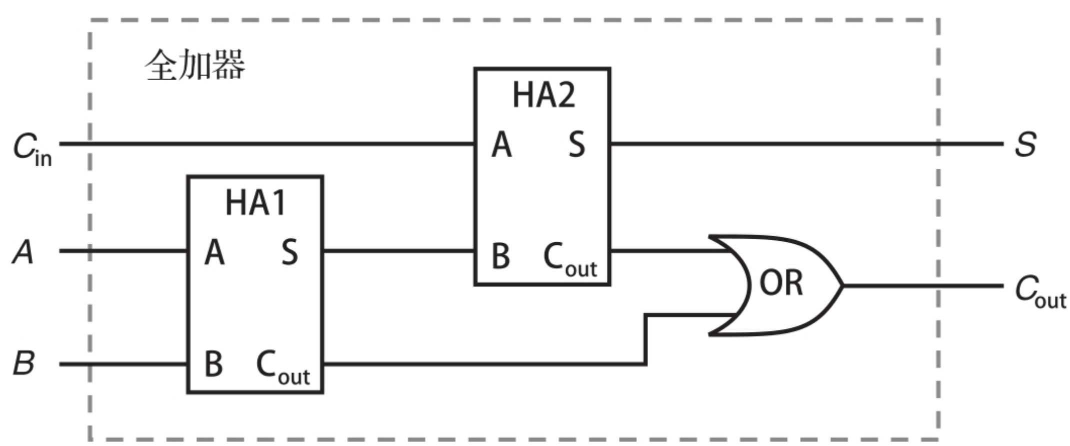
为了全面了解全加器可能的输入和输出，我们可以使用真值表，如表5-2所示。这个表有3个输入（*A*、*B*、*C*）和2个输出（*S*、*C*）。花点时间考虑一下各种输入组合对应的输出。

inout

怎样实现全加器呢？顾名思义，全加器可以通过两个半加器来实现（见图5-13）。

▼表5-2 全加器真值表

输入			输出	
A	*B*	C_{in}	*S*	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



▲图5-13 用两个半加器和一个OR门实现全加器

全加器的和数位输出（*S*）应该是*A*和*B*的和（可以用一个半加器HA1计算）再加上*C*（可以用第二个半加器HA2计算），如图5-13所示。

in

全加器还需要输出进位位。事实证明，这实现起来很简单，因为如果任何一个半加器的进位为1，那么全加器中*C*的值就是1。因此，我们可以用一个OR门来实现，如图5-13所示。

out

这也是封装的一个例子。电路构造好后，就可以在不知道具体实现细节的情况下使用全加器功能了。下一节将介绍如何使用全加器和半加器来实现多位数的加法。

5.4 4位加法器

全加器允许我们执行两个1位数再加上一个进位位的加法。这为我们提供了搭建电路的构建块，使电路能进行多个位的二进制数加法。现在，我们把几个1位加法器电路组合起来构成一个4位加法器。最低有效位用半加器（因为它不需要进位位），其他位用全加器。如图5-14所示，我们把加法器串在一起，这样每个加法器的进位输出就会接入后续加法器的进位输入端。

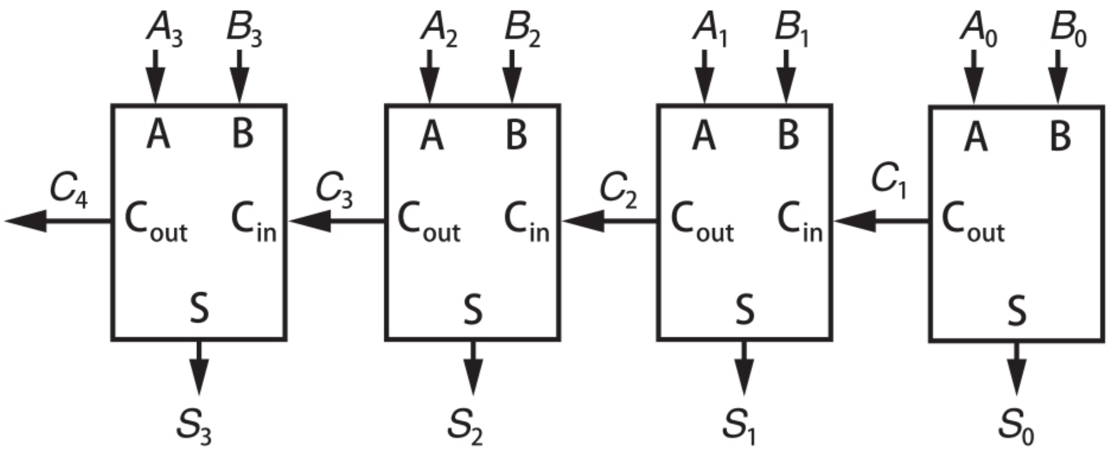


图5-14 4位加法器

为了和人们书写数字的方式一致，图5-14把最低有效位放在右边，且计算流程是从右到左的。这就意味着我们的加法器框图的输入和输出位置将和前面展示的不同，不要让它迷惑了你！

在图5-15中，我用这个4位加法器重新计算了之前的例子：0010+0011。

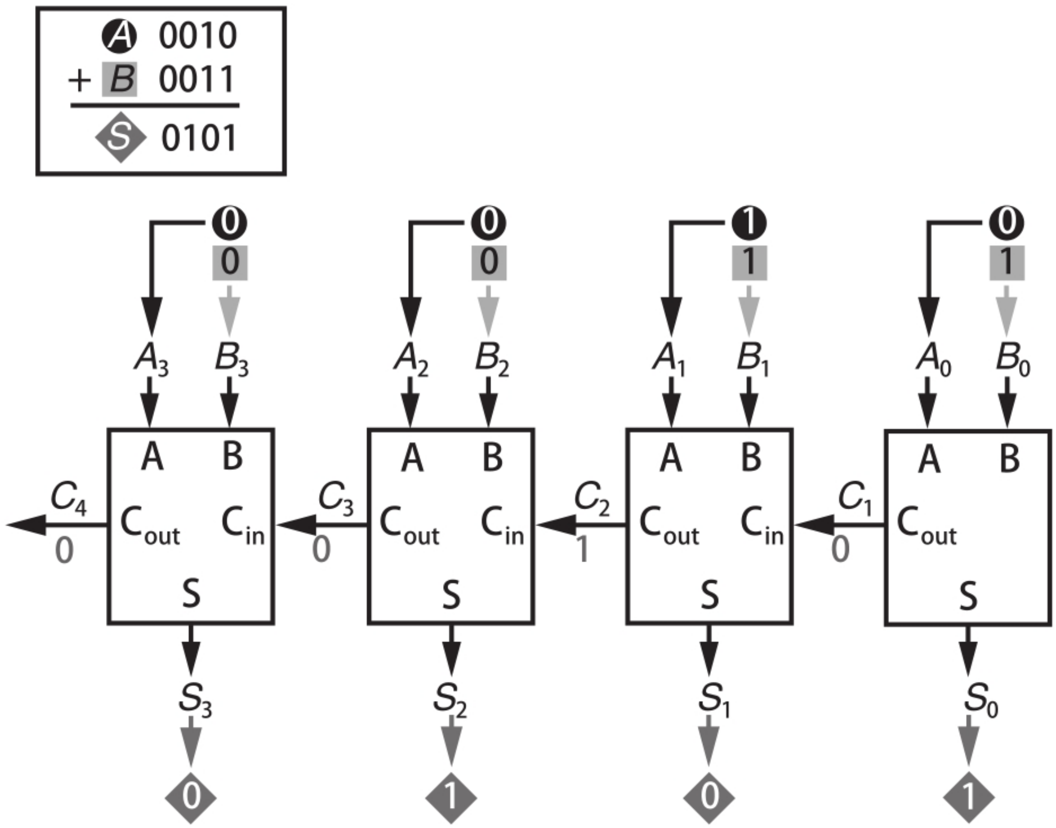


图5-15 4位加法器的工作过程

在图5-15中，我们可以看到输入*A*（0010）和输入*B*（0011）是如何被送入每个加法器单元的，从右边的最低有效位开始，一直移动到左边的最高有效位。你可以按从右到左的顺序来处理图中的计算流程。首先将最右边的0（*A*）和1（*B*）相加，结果为1（*S*），进位为0。

000

最右边加法器的进位输出为*C*，接入下一个加法器，这个加法器会将1（*A*）和1（*B*）以及进位0相加。结果为0（*S*）和进位1（*C*）。继续这个过程，直到最左边的加法器完成运算为止。最终结果是一组输出位0101（*S*~*S*）和一个进位0（*C*）。如果需要处理更多的位，那么可以通过合并更多全加器来扩展图5-15中的设计。

11112304