$\sim x_{k+1}$，1，0，…，0]。也就是，我们对位 k 左边的所有位取反。

我们用一些 4 位数字来说明这个方法，这里我们用斜体来突出最右边的模式 1，0，…，0：

x		$-x$	
[1*100*]	−4	[0*100*]	4
[*1000*]	−8	[*1000*]	−8
[010*1*]	5	[101*1*]	−5
[011*1*]	7	[100*1*]	−7

2.3.4 无符号乘法

范围在 $0 \leqslant x$，$y \leqslant 2^w - 1$ 内的整数 x 和 y 可以被表示为 w 位的无符号数，但是它们的乘积 $x \cdot y$ 的取值范围为 0 到 $(2^w-1)^2 = 2^{2w} - 2^{w+1} + 1$ 之间。这可能需要 $2w$ 位来表示。不过，C 语言中的无符号乘法被定义为产生 w 位的值，就是 $2w$ 位的整数乘积的低 w 位表示的值。我们将这个值表示为 $x *^{\mathrm{u}}_{w} y$。

将一个无符号数截断为 w 位等价于计算该值模 2^w，得到：

原理：无符号数乘法

对满足 $0 \leqslant x$，$y \leqslant UMax_w$ 的 x 和 y 有：

$$x *^{\mathrm{u}}_{w} y = (x \cdot y) \bmod 2^w \tag{2.16}$$

2.3.5 补码乘法

范围在 $-2^{w-1} \leqslant x$，$y \leqslant 2^{w-1} - 1$ 内的整数 x 和 y 可以被表示为 w 位的补码数字，但是它们的乘积 $x \cdot y$ 的取值范围为 $-2^{w-1} \cdot (2^{w-1}-1) = -2^{2w-2} + 2^{w-1}$ 到 $-2^{w-1} \cdot -2^{w-1} = -2^{2w-2}$ 之间。要想用补码来表示这个乘积，可能需要 $2w$ 位。然而，C 语言中的有符号乘法是通过将 $2w$ 位的乘积截断为 w 位来实现的。我们将这个数值表示为 $x *^{\mathrm{t}}_{w} y$。将一个补码数截断为 w 位相当于先计算该值模 2^w，再把无符号数转换为补码，得到：

原理：补码乘法

对满足 $TMin_w \leqslant x$，$y \leqslant TMax_w$ 的 x 和 y 有：

$$x *^{\mathrm{t}}_{w} y = U2T_w((x \cdot y) \bmod 2^w) \tag{2.17}$$

我们认为对于无符号和补码乘法来说，乘法运算的位级表示都是一样的，并用如下原理说明：

原理：无符号和补码乘法的位级等价性

给定长度为 w 的位向量 $\vec{x}$ 和 $\vec{y}$，用补码形式的位向量表示来定义整数 x 和 y：$x = B2T_w(\vec{x})$，$y = B2T_w(\vec{y})$。用无符号形式的位向量表示来定义非负整数 x' 和 y'：$x' = B2U_w(\vec{x})$，$y' = B2U_w(\vec{y})$。则

$$T2B_w(x *^{\mathrm{t}}_{w} y) = U2B_w(x' *^{\mathrm{u}}_{w} y')$$

作为说明，图 2-27 给出了不同 3 位数字的乘法结果。对于每一对位级运算数，我们执行无符号和补码乘法，得到 6 位的乘积，然后再把这些乘积截断到 3 位。无符号的截断后的乘积总是等于 $x \cdot y \bmod 8$。虽然无符号和补码两种乘法乘积的 6 位表示不同，但是截断后的乘积的位级表示都相同。

推导：无符号和补码乘法的位级等价性

根据等式(2.6)，我们有 $x' = x + x_{w-1}2^w$ 和 $y' = y + y_{w-1}2^w$。计算这些值的乘积模 2^w 得到以下结果：

$$(x' \cdot y') \bmod 2^w = [(x + x_{w-1}2^w) \cdot (y + y_{w-1}2^w)] \bmod 2^w$$