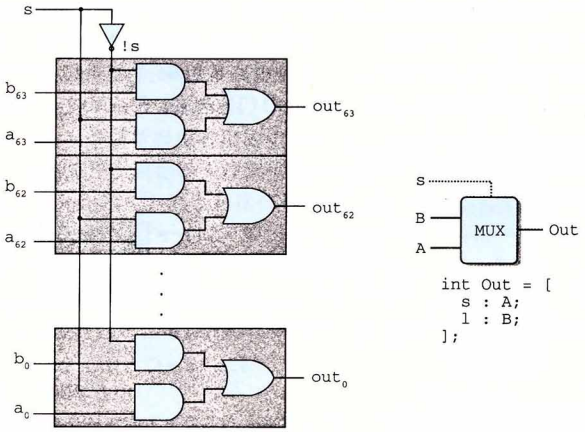
这个表达式包含一系列的情况，每种情况 i 都有一个布尔表达式 $select_i$ 和一个整数表达式 $expr_i$，前者表明什么时候该选择这种情况，后者指明的是得到的值。



a）位级实现　　b）字级抽象

图 4-13　字级多路复用器电路。当控制信号 s 为 1 时，输出会等于输入字 A，否则等于 B。HCL 中用情况(case)表达式来描述多路复用器

同 C 的 switch 语句不同，我们不要求不同的选择表达式之间互斥。从逻辑上讲，这些选择表达式是顺序求值的，且第一个求值为 1 的情况会被选中。例如，图 4-13 中的字级多路复用器用 HCL 来描述就是：

```
word Out = [
        s: A;
        1: B;
];
```

在这段代码中，第二个选择表达式就是 1，表明如果前面没有情况被选中，那就选择这种情况。这是 HCL 中一种指定默认情况的方法。几乎所有的情况表达式都是以此结尾的。

允许不互斥的选择表达式使得 HCL 代码的可读性更好。实际的硬件多路复用器的信号必须互斥，它们要控制哪个输入字应该被传送到输出，就像图 4-13 中的信号 s 和!s。要将一个 HCL 情况表达式翻译成硬件，逻辑合成程序需要分析选择表达式集合，并解决任何可能的冲突，确保只有第一个满足的情况才会被选中。

选择表达式可以是任意的布尔表达式，可以有任意多的情况。这就使得情况表达式能描述带复杂选择标准的、多种输入信号的块。例如，考虑图 4-14 中所示的四路复用器的图。这个电路根据控制信号 s1 和 s0，从 4 个输入字 A、B、C 和 D 中选择一个，将控制信号看作一个两位的二进制数。我们可以用 HCL 来表示这个电路，用布尔表达式描述控制位模式的不同组合：

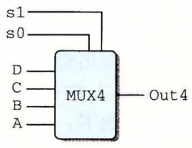


图 4-14　四路复用器。控制信号 s1 和 s0 的不同组合决定了哪个数据输入会被传送到输出

```
word Out4 = [
        !s1 && !s0 : A; # 00
```