

这段代码简单地定义了位级(数据类型 `bool` 表明了这一点)信号 `eq`, 它是输入 `a` 和 `b` 的函数。从这个例子可以看出 HCL 使用了 C 语言风格的语法, ‘=’ 将一个信号名与一个表达式联系起来。不过同 C 不一样, 我们不把它看成执行了一次计算并将结果放入内存中某个位置。相反, 它只是给表达式一个名字。


 **练习题 4.9** 写出信号 `xor` 的 HCL 表达式, `xor` 就是异或, 输入为 `a` 和 `b`。信号 `xor` 和上面定义的 `eq` 有什么关系?

图 4-11 给出了另一个简单但很有用的组合电路, 称为多路复用器(multiplexor, 通常称为“MUX”)。多路复用器根据输入控制信号的值, 从一组不同的数据信号中选出一个。在这个单个位的多路复用器中, 两个数据信号是输入位 `a` 和 `b`, 控制信号是输入位 `s`。当 `s` 为 1 时, 输出等于 `a`; 而当 `s` 为 0 时, 输出等于 `b`。在这个电路中, 我们可以看出两个 AND 门决定了是否将它们相对应的数据输入传送到 OR 门。当 `s` 为 0 时, 上面的 AND 门将传送信号 `b`(因为这个门的另一个输入是 `!s`), 而当 `s` 为 1 时, 下面的 AND 门将传送信号 `a`。接下来, 我们来写输出信号的 HCL 表达式, 使用的就是组合逻辑中相同的操作:

```
bool out = (s && a) || (!s && b);
```

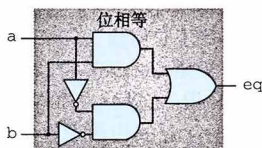


图 4-10 检测位相等的组合电路。当输入都为 0 或都为 1 时, 输出等于 1

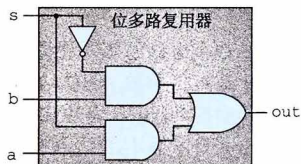


图 4-11 单个位的多路复用器电路。如果控制信号 `s` 为 1, 则输出等于输入 `a`; 当 `s` 为 0 时, 输出等于输入 `b`

HCL 表达式很清楚地表明了组合逻辑电路和 C 语言中逻辑表达式的对应之处。它们都是用布尔操作来对输入进行计算的函数。值得注意的是, 这两种表达计算的方法之间有以下区别:

- 因为组合电路是由一系列的逻辑门组成, 它的属性是输出会持续地响应输入的变化。如果电路的输入变化了, 在一定的延迟之后, 输出也会相应地变化。相比之下, C 表达式只会在程序执行过程中被遇到时才进行求值。
- C 的逻辑表达式允许参数是任意整数, 0 表示 FALSE, 其他任何值都表示 TRUE。而逻辑门只对位值 0 和 1 进行操作。
- C 的逻辑表达式有个属性就是它们可能只被部分求值。如果一个 AND 或 OR 操作的结果只用对第一个参数求值就能确定, 那么就不会对第二个参数求值了。例如下面的 C 表达式:

```
(a && !a) && func(b,c)
```

这里函数 `func` 是不会被调用的, 因为表达式 `(a && !a)` 求值为 0。而组合逻辑没有部分求值这条规则, 逻辑门只是简单地响应输入的变化。

#### 4.2.3 字级的组合电路和 HCL 整数表达式

通过将逻辑门组合成大的网, 可以构造出能计算更加复杂函数的组合电路。通常, 我们设计能对数据字(word)进行操作的电路。有一些位级信号, 代表一个整数或一些控制模