式。例如，我们的处理器设计将包含有很多字，字的大小的范围为 4 位到 64 位，代表整数、地址、指令代码和寄存器标识符。

执行字级计算的组合电路根据输入字的各个位，用逻辑门来计算输出字的各个位。例如图 4-12 中的一个组合电路，它测试两个 64 位字 A 和 B 是否相等。也就是，当且仅当 A 的每一位都和 B 的相应位相等时，输出才为 1。这个电路是用 64 个图 4-10 中所示的单个位相等电路实现的。这些单个位电路的输出用一个 AND 门连起来，形成了这个电路的输出。

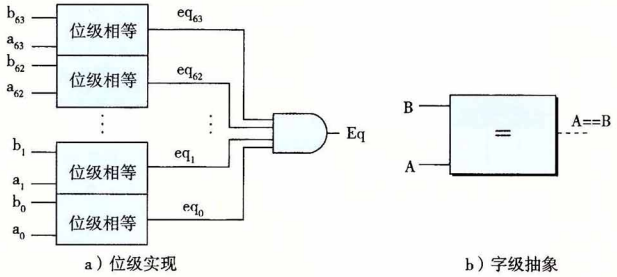

图 4-12　字级相等测试电路。当字 A 的每一位与字 B 中相应的位均相等时，输出等于 1。字级相等是 HCL 中的一个操作

在 HCL 中，我们将所有字级的信号都声明为 int，不指定字的大小。这样做是为了简单。在全功能的硬件描述语言中，每个字都可以声明为有特定的位数。HCL 允许比较字是否相等，因此图 4-12 所示的电路的函数可以在字级上表达成

```
bool Eq = (A == B);
```

这里参数 A 和 B 是 int 型的。注意我们使用和 C 语言中一样的语法习惯，'= '表示赋值，而'=='是相等运算符。

如图 4-12 中右边所示，在画字级电路的时候，我们用中等粗度的线来表示携带字的每个位的线路，而用虚线来表示布尔信号结果。

练习题 4.10　假设你用练习题 4.9 中的异或电路而不是位级的相等电路来实现一个字级的相等电路。设计一个 64 位字的相等电路需要 64 个字级的异或电路，另外还要两个逻辑门。

图 4-13 是字级的多路复用器电路。这个电路根据控制输入位 s，产生一个 64 位的字 Out，等于两个输入字 A 或者 B 中的一个。这个电路由 64 个相同的子电路组成，每个子电路的结构都类似于图 4-11 中的位级多路复用器。不过这个字级的电路并没有简单地复制 64 次位级多路复用器，它只产生一次!s，然后在每个位的地方都重复使用它，从而减少反相器或非门(inverters)的数量。

处理器中会用到很多种多路复用器，使得我们能根据某些控制条件，从许多源中选出一个字。在 HCL 中，多路复用函数是用情况表达式(case expression)来描述的。情况表达式的通用格式如下：

```
[
    select_1  :  expr_1;
    select_2  :  expr_2;
    .
    .
    .
    select_k  :  expr_k;
]
```