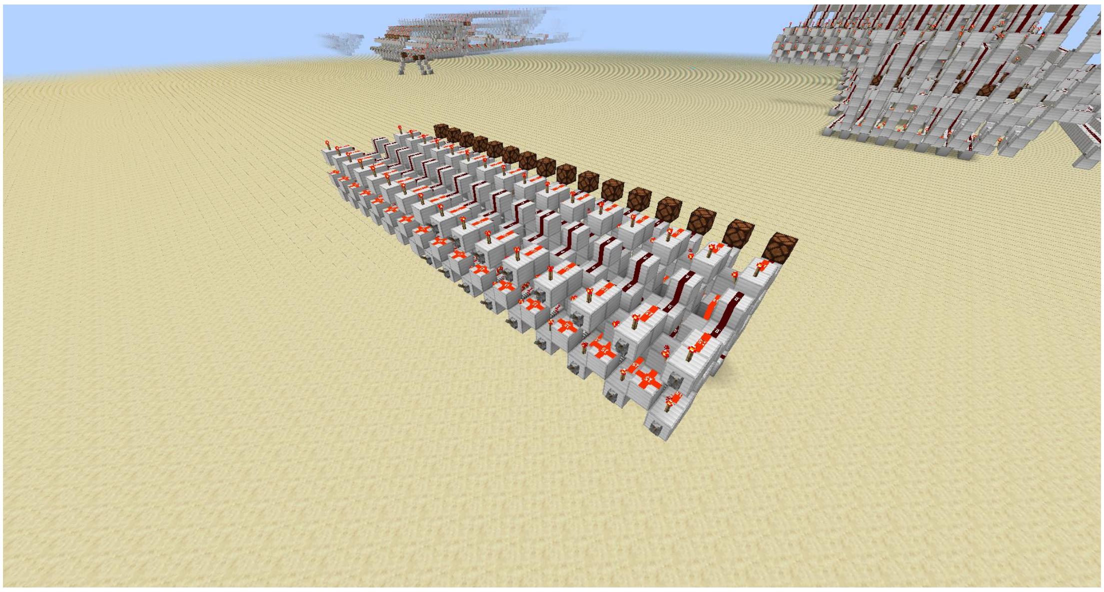
更高效的乘法器——树状乘法器原理与建造

2021 年 12 月 26 日

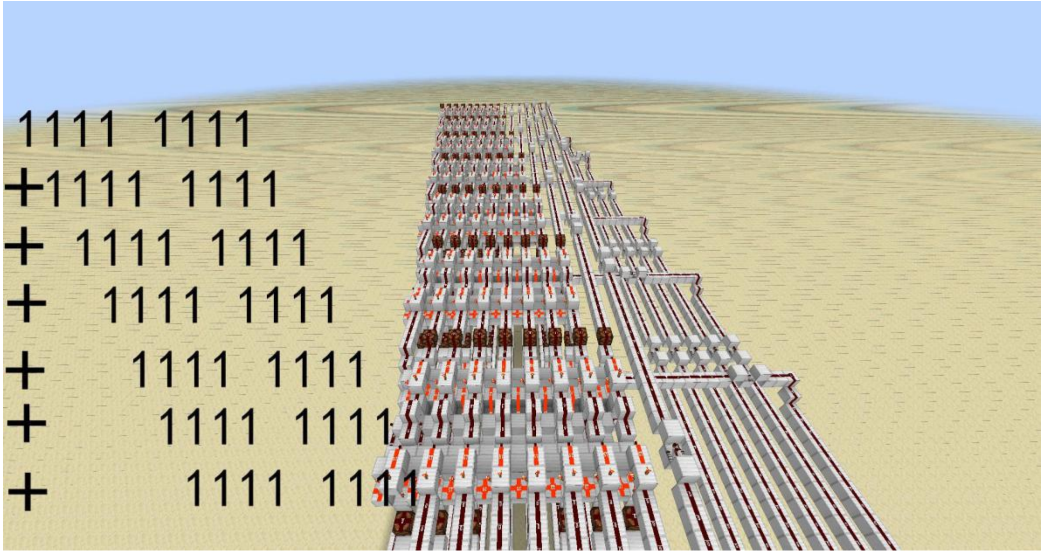
By Enity_303-E3

我们知道，乘法的本质是加法，而加法需要用到全加器，因此这里先单独展示16位全加器的构造。具体的建造方式因篇幅限制这里不多赘述，读者可自行上网查阅相关资料。



（上图：单个16位全加器）

这里先举个例子，110*101 可以将乘数 101 拆为 1,0,1，每一位与被乘数 110 相乘，即进行与运算[①]，然后依次向高位移一位，结果分别为 110,0000,11000，最后将这些结果加起来， 得到最终结果 11110。一般乘法器就是将这些结果累加起来，得到最终结果。这里也不多赘述一般乘法器的建造过程，只展示整体构造。



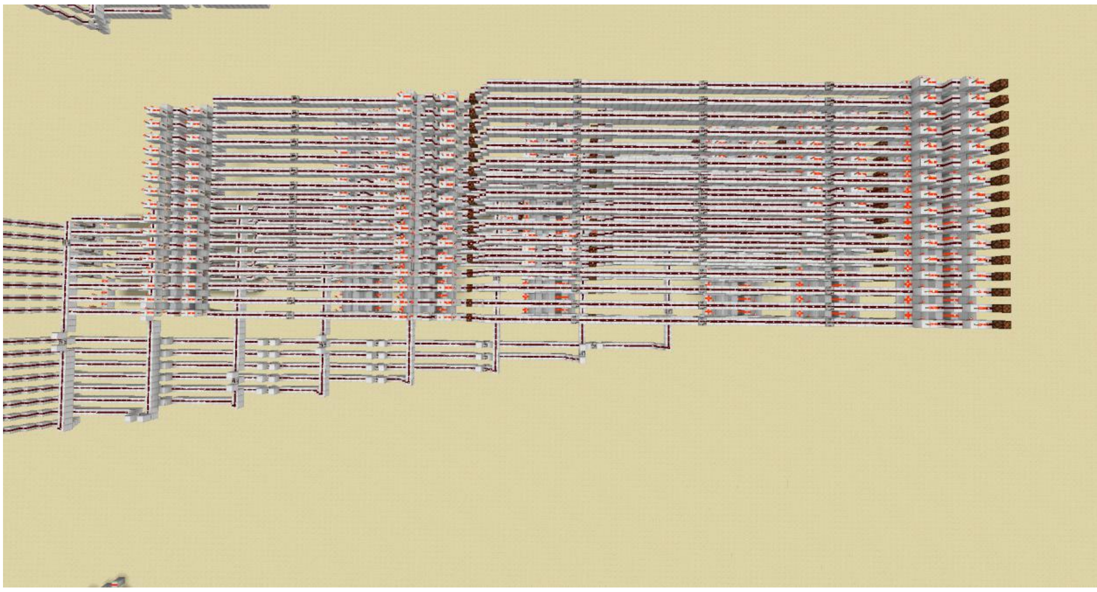
（上图：一般乘法器）

由一般乘法器原理容易得到，一个n 位的乘法器（即最大输入 2ˆ(n-1)＊2ˆ(n-1)） 总共需要 n-1 次加法运算。设全加器计算复杂度为 O(1)[2]，若不考虑移位与信号传输延时，则 n 位乘法器计算复杂度为 O(n-1)。例如， 1111 1111*1111 1111，一般乘法器需要 7 次加法（即 1111 1111+1 1111 1110+......+111 1111 1000 0000）。显然， 对于高位（如 32 或 64 位）的乘法计算复杂度就很大。

因此，我们有一种更高效的乘法器——树状乘法器。我们不妨先把第 1,2 位，第 3,4 位，第 5,6 位，第 7,8 位分别相加，这四次加法运算不考虑移位与信号传输延迟是完全可以实现并行[3]的，因此这一轮加法计算复杂度为 O(1)。同理，将以上结果再两两相加，不断重复只要最后一个结果。不难得出，总体计算复杂度为 O(log n[4])。这就是树状乘法器的原理。而这两两相加的过程，我们能联想到“二叉树”，所以这种乘法器由此得名“树状乘法器”。

以下介绍树状乘法器的具体建造方式。

树状乘法器的整体概览如下图所示。



这里以 8 位*8 位为例。首先是一般乘法器的移位（这里不多做解释）。我们不妨将8次移位后的结果命名为a1,a2,a3,a4,a5,a6,a7,a8。

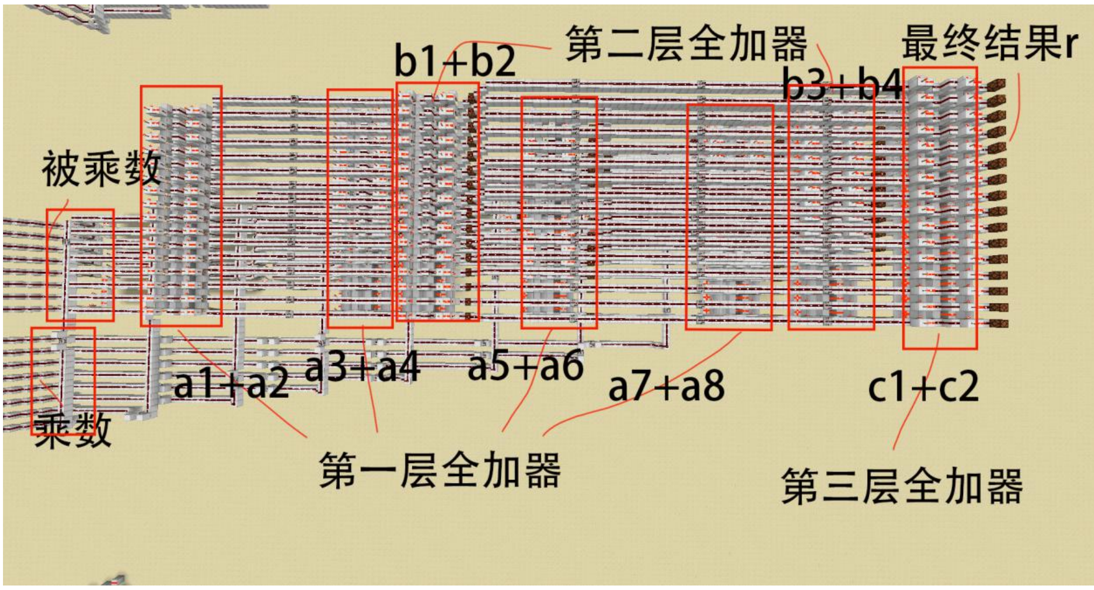
第一步：将a1与a2，a3与a4，a5与a6，a7与a8分别使用全加器相加，得到结果b1，b2，b3，b4，这一步计算复杂度为O(1)

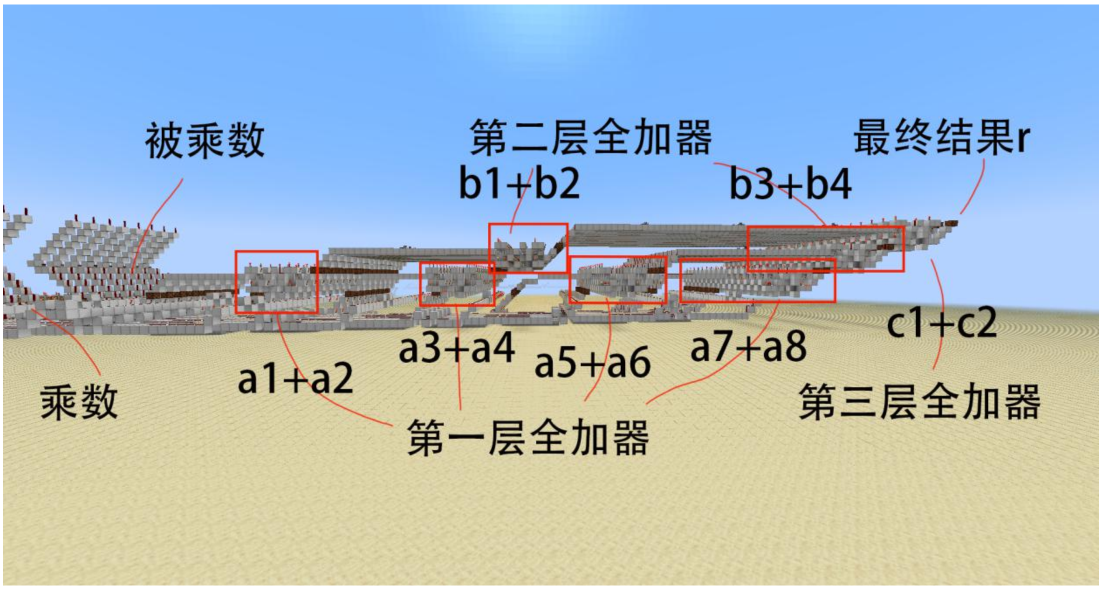
第二步：将b1与b2，b3与b4分别使用全加器相加，得到结果c1,c2，这一步计算复杂度为O(1)。

第三步：将c1与c2使用全加器相加得到结果r，这一步计算复杂度为O(1)。

r就是最终结果，总计算复杂度位O(3)。

总体布局的俯视图与侧视图如下图所示。


第二层全加器
b1+b2
最终结果r
b3+b4
被乘数
a1+a2
a3+a4
a5+a6
a7+a8
c1+c2
乘数
第一层全加器
第三层全加器


被乘数
第二层全加器
最终结果r
b1+b2
b3+b4
c1+c2
a3+a4
a5+a6
a7+a8
a1+a2
乘数
第三层全加器
第一层全加器

树状乘法器有着更低的计算复杂度，并且建造难度较低，此外配合cca（封闭进位加法器）可以获得更好的效果，这是红石计算器建造中可供选择的方案之一。

最后感谢辰占鳌头与[mail_set@arch ~]$的提醒与指导。

注释：①与运算：即逻辑与运算（&&或and），1&&1=1，0&&0=0&&1=1&&0=0。举例中110*1即110&&1=110，110*0即110&&0=000。②计算复杂度O(n)（注意是英语字母O不是数字零），代表计算所需用时为n。③并行：即在同一时刻完成多个任务。④这里的log n是以2为底数的，即$\log_2 n$

参考文献

无参考文献