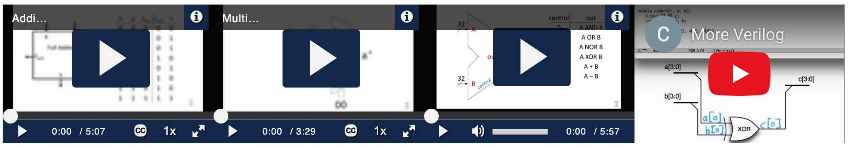
维迪 操作系统



视频来源：Geoffrey Herman 和 Craig Zilles，文字来源：Geoffrey Herm 一个

大图 尤瑞

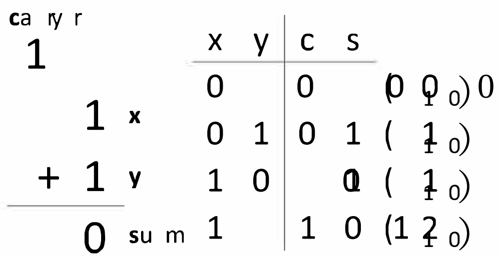
计算机可以做两件事：存储状态和操纵状态。操纵状态的主要机制是算法 埃蒂 c
逻辑单元。

算术逻辑单元是处理器的主要部分。它执行所有算术运算和位逻辑 艾尔
我们到目前为止教给您的操作。要了解 ALU 的工作原理，您需要了解数据和控制的概念。数据是我们正在处理的信息，而控制告诉我们在处理该数据时要执行哪些操作。 佛 r
例如，ALU 负责实现 C 代码，如 X = A + B; 或 X = A & B;。ALU 接收两部分数据（例如，A 和 B）并产生一个新数据（例如，X）。我们使用控制位来决定执行哪个操作（例如 + 或者 &)。为了帮助您识别何时将线路视为数据或控制，我们将使用红色表示数据，使用蓝色表示控制。

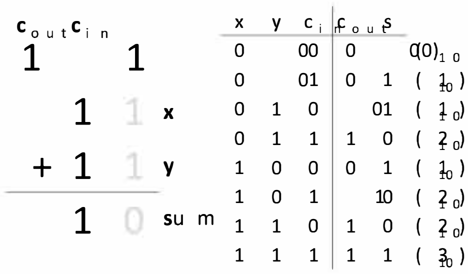
为了构建 ALU，我们使用模块化设计。我们设计了一个 1 位算术单元和一个 1 位逻辑单元。然后我们将它们链接起来 塞
电路连接在一起。

一位 sli 的二进制加法电路 行政长官

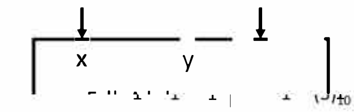
如果我们想将两个 N 位二进制数相加，通常从每个数的最低有效位开始。如果我们将这些 埃
两位加在一起，我们有三种可能的结果：它们的和是 0 1 0、1 1 0 或 2 1 0。因为我们无法用 时长
一位，我们将加法中的最高有效位移到下一位位置。这个过程表明我们需要两个输出 乌特
位来表示这些位的总和：一个和位和一个进位位。我们可以表示实现 日 是 特斯
加法运算，如下所示。我们称该真值表所表示的电路为半加法器，因为它 s
仅完成加法工作的一半。



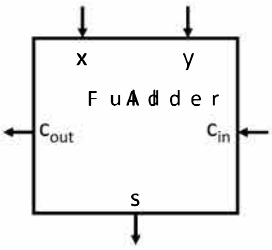
如果我们继续将加法运算到下一个位置，我们就会意识到必须将三个位相加：x、y 和 cin。因此，此操作可以输出二进制数 0-3。幸运的是，我们仍然可以用以下代码对这些数字进行编码 台灣o
位 - 我们的和与进位位。我们可以用以下真值表来表示实现此加法运算的电路的行为。我们将此真值表所表示的电路称为全加器。



我们可以使用逻辑门来实现这个电路。我们通常将此电路封装为具有以下功能的模块 翼
外观，通常使用全加器的缩写FA。

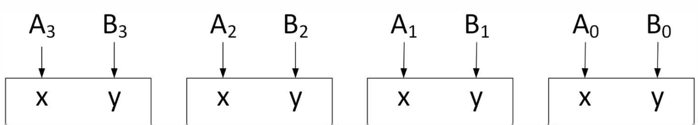


我们可以使用逻辑门来实现这个电路。我们通常将此电路封装为具有以下功能的模块 翼
外观，通常使用全加器的缩写FA。



N位二进制加法 圈 公寓

为了执行 N 位二进制加法，我们使用模块化设计，将 N 个全加器链接在一起。我们需要将 0 输入到 埃
最低有效位的全加器。例如，下面显示了 4 位加法和 8 位加法电路。



PRE 3

Assessment overview 看法

Total points: 0 / 75

Score: 0%

问题 PRE0 3.1

价值：1 G 总积分：

- /1

自动评分 问题 d

tion IHi+

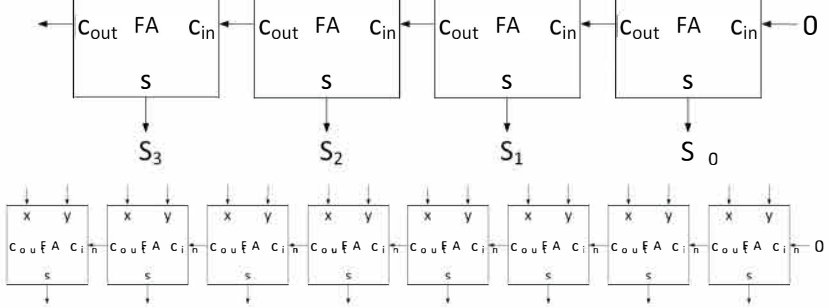
HE

的 不是 西文

No attached files

Attach a file

Add text note



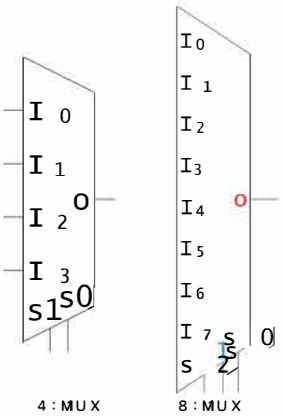
多路复用 （多路复用器

硬件中常见的设计模式是并行计算许多不同的值，然后使用多路复用器 (MUX) 仅选择我们需要的值。我们将要从数据输入中选择的值集合称为接收 否 西文

数据输入将需要 log2N 个选择输入位来编码要选择的数据输入。我们用十进制标记数据输入 升

以零索引开头的数字下标（例如，I_0 或 I_2）应解释为十进制数。我们用下标（例如，s_2 或 s0）标记选择输入，这些下标对应于 N 位无符号二进制位置的指数 注释 仿

（即 $s_2\ s_1\ s0$ 对应 $s_2*2^2+ s1*2^1+ s0*2^0$），其中 N 为选择位数。例如，如果 $s_2\ s_1\ s0 =$ 101，则应该 被解释为 5（即 $1*2^2+ 0*2^1+ 1*2^0$），这表示应该选择数据输入 Is。



4：MUX behavior table

s_1	s_0	o
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

8：MUX behavior table

s_2	s_1	s_0	o
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

总线和 N 位宽 MUXe s

为了表示 N 位信号（如我们的 N 位加法器），我们将电线捆扎成称为总线的组。总线通常表示 总线表示 d

为一条带有斜线的导线，并标明该总线中的导线数量。其他时候，总线表示为更粗的 金属线

电路图。当电线捆绑为总线时，我们使用数组符号来指示电线的位位置。F 或者

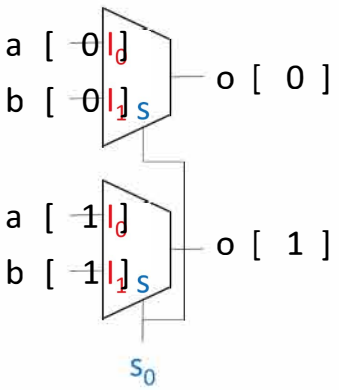
例如，如果我们在总线 A 上存储了一个 4 位无符号二进制数 1011，那么 A[31= = 1、A[21= = 0、A[11= = 1、A[01= = 1。如果需要我们，可以通过将总线 A 的大小写为 A[3:0] 来指示它。Obit 位置始终被解释为最低有效位 位位置。 蚂蚁

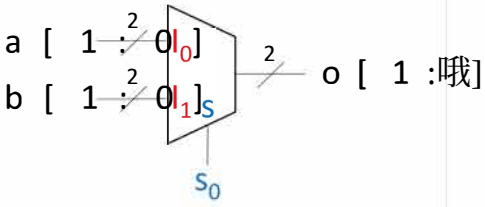
在执行按位逻辑运算或其他多位运算等操作时，我们可能希望对总线上的每条线路执行相同的操作。例如，在下图中，我们对多个 MUX 使用单个选择输入位

这样它们要么选择所有 a 输入，要么选择所有 b 输入（绝不会混合搭配）。这对于 2 位 信号 纳尔

但对于位数更多的信号，绘制起来很麻烦，所以我们使用像右边这样的简写，其中我们显示整个总线进入单个 MUX。虽然原理图只显示一个 MUX，但实际上我们的电路将有 N 个 MUX。我们称之为 N 位宽 MUX。每当我们将这个简写与门或 MUX 一起使用时，我们都表示我们正在对每个执行相同的操作 双 吨

在公交车上切片。





1 位逻辑 统一 吨

逻辑单元是一个模块，让我们可以在 L 个按位逻辑运算之间进行选择。我们通过创建来实现逻辑单元 1-b 它

逻辑单元，然后多次复制该模块以形成 N 位逻辑单元。例如，我们将使用 32 位逻辑 单元

对 32 位 int 变量执行按位逻辑运算（例如 int A 和 int B）。

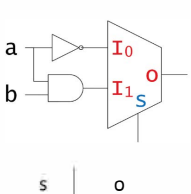
我们通过对来自每个 o 的 1 位执行我们希望执行的每个按位逻辑运算来实现 1 位逻辑单元 日 埃

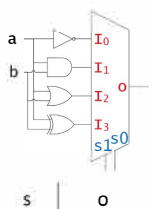
两个输入变量（例如，分别来自 A 和 B 的 a[0] 和 b[0]），并选择我们实际想要使用的

多路复用器（见下图）。这种设计可能有点违反直觉（计算所有的按位逻辑不是很浪费吗？ 卡尔

操作只是为了扔掉它们中的大多数？），但这种并行执行许多计算的设计模式忽略了 莫斯 吨

其中是在硬件中执行计算的最快方法。





0	NOT a
1	a AND b

0 0	NOT a
0 1	a AND b
1 0	a OR b
1 1	a XOR b

逻辑单元可以具有任意大小或布尔运算顺序。当我们朝着实现 MIPS 数据结构的方向发展时 我们 小时，
将使用以下实现来实现逻辑单元。

s	o
0 0	a AND b
0 1	a OR b
1 0	a NOR b
1 1	a XOR b

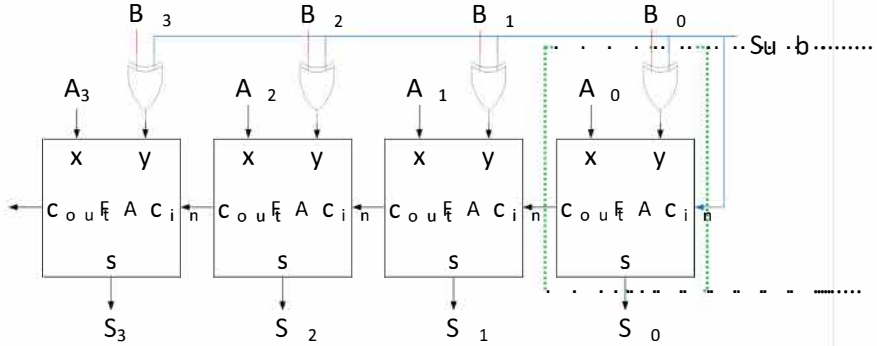
算术逻辑单元（AL （U）

计算机的计算能力是算术逻辑单元 (ALU)。ALU 结合了几个较小的模块 —富勒 升
加法器、异或门、多路复用器和逻辑单元——创建一个电路，让我们在一组基本算术之间进行选择 操作 离子
（加法和减法）和按位逻辑运算（AND、OR、NOR、XOR）。我们将设计一个 1 位 ALU，然后链接这些 埃
ALU 组合在一起，形成 N 位 ALU。

算术 统一 吨知道了

下面是一个4位算术单元，可以执行4位加法和减法。Sub是一个控制信号，使电路 伊特

下面是一个4位算术单元，可以执行4位加法和减法。Sub是一个控制信号，使电路 吨
当为 0 时执行加法，当为 1 时执行减法。当 Sub= = 0 时，XOR 门不对 B_1 信号进行补码 所以 挨
加法器执行 A+ B。当 Sub= = l 时，XOR 门按位补充 B_1 信号，以便加法器执行 A+ ~ B。此外， 日 埃
Sub 信号是最低有效位 C_1 n 的输入，当 Sub= = l 时，它会加 1。当 Sub= = l 时，加法器一起执行 A+ ~ B+ 升
= A+（ ~ B+ l） = A-
B。

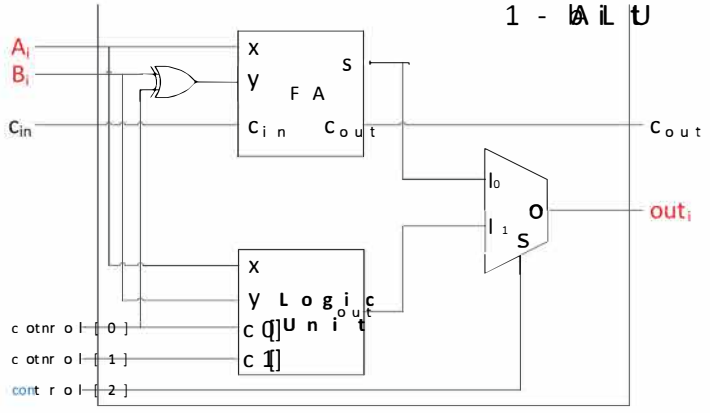


我们可以只使用全加器和异或门（绿色虚线框）来制作 1-bi 算术逻辑单元

1 位算术 逻辑 统一 吨

为了构建 1 位 ALU，我们使用 2:1 MUX 将 1 位算术单元和 1 位逻辑单元组合在一起。这只是 1-
位 ALU。有很多不同的可能性，但这是我们在本课中将使用的一种。这个 1 位 ALU 使用 1 位逻辑单元 吨
使用以下行为表。

c[1:0]	b
0 0	a AND b
0 1	a OR b
1 0	a NOR b
1 1	a XOR b



通过使用 MUX 将算术单元与逻辑单元组合，此 1 位 ALU 的行为表将是以下内容 翼。
请注意，每个 1 位 ALU 并不单独执行减法，而是执行 a_i 加上 b_i 的补码。 日 城
额外的“ + 1 ”，减法不完整。“ + 1 ”仅发生在 N 位 ALU 级别。

control[2:0]	out
0 0 0	undefi 内德
0 0 1	undefi 内德
0 1 0	a_i b $_i$
0 1 1	a_i + ~ b_i
1 0 0	a_i b $_i$ N 德
1 0 1	a O Rb $_i$
1 1 0	a NOR b_i
1 1 1	a XOR b_i

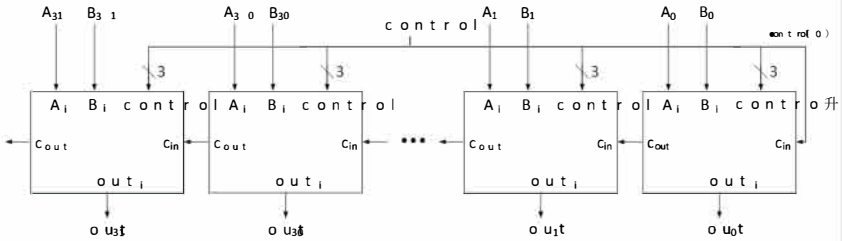
在此行为表中，未定义表示电路执行的任何行为都是可以的。这并不意味着电路 有一个
不确定的输出。如果你仔细跟踪电路，你会看到控制= = 000 也实现了加法，而控制= 0 艾尔斯01

实现减法。这种冗余是可以的，因为我们没有定义该行为。

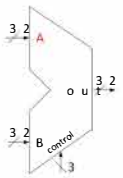
N位A 鲁

我们通过将 N 个 1 位 ALU 链接在一起来构建 N 位 ALU。下图显示了一个 32 位 ALU。请注意每个 1 位 ALU 每个 1 位 ALU 都从总线 A[31:0] 和 B[31:0] 接收不同的位，但每个 1 位 ALU 都接收相同的控制信号。通过给每个 1 位 ALU 具有相同的控制信号，每个 1 位 ALU 作为整体的一部分执行相同的操作。

注意控制 [0] 是如何发送到最低有效位的 Cin 的，就像我们在算术中对 Sub 信号所做的那样 统一 哦知道了 当 Cin==1 时，它提供 A + (~ B + 1) 中的 “ + 1 ” 来实现减法。Cin 是 ALU 的算术单元部分的 Sub 。



我们不画出所有 32 个 1 位 ALU，而是画出具有以下形状的 ALU，并带有总线 秒。



在此示例中，32 位 ALU 具有以下行为 标签 r勒。

control[2:0]	输出
000	undefi 内德
001	undefi 内德

补充可选阅读 韓：

Mano & Kime 第 4 版，4. 3

标记为真实 d

0（a）我读过 是！

选择所有适用的可能选项 格 。

保存并评分 Unlimitedattempts 保存 仅有的