计算机体系之变迁

（征询意见稿）

第二分册

张　修编

中国科学院计算技术服务社

一九八一年七月

第二分册目录

第三章 从百花齐放到统一系列

（单机体系结构的发展）

从1954年贝尔实验室制成第一台晶体管计算机TRADIC开始，进入了第二代计算机阶段。这是计算机大发展的阶段。晶体管给计算机带来了廉价可靠的特点。计算机应用范围迅速扩大，从象牙之塔走上了市井街头。体系设计者思想活跃，提出了不少新颖而奇特的方案。其中著名的有：

UNIVAC LARC是1960年斯别利·兰德公司安装的一台高速通用计算机。它除了用于计算的主机以外，还安装了一个控制输入输出的独立的外围处理机。成为第一台采用多处理机工作方式的系统。它也是首批使用多道程序的计算机。在本章第五节将做介绍。

IBM STRETCH（即IBM7030）是IBM公司在1961年制成的著名的晶体管计算机。它吸收了当时许多成功的设计思想，也提出了一些新思想和新概念。体系结构和位组等概念就是在设计它时首次提出来的。它采用了先行控制工作方式。这种方式以后发展成流水线方式。所以它被认为是第一台流水线计算机。它还采用交换去控制外部设备的操作，这种方式以后发展成通道控制。这台机器既装有浮点运算器，也装有十进制运算器，它为以后IBM 360和370系列的通用机结构奠定了基础。但是它的功能过于庞大，控制繁杂，因而在市场上没有取得预想的效果。本章第二节将做介绍。

ATLAS是1962年英国曼彻斯特大学设计的一台计算机。它以首

次提出虚拟存储器（当时称为一次存储器）而著称。本章第四节将介绍。

CDC 6600是1964年控制数据公司制成的主要用于科学计算的计算机。它采用了多部件并行计算及主机与外围处理机并行工作的方式。该公司在1969年制成的7600是这些设计思想的进一步发展。这两台计算机将在本章第三节中介绍。

B 5500是巴勒斯公司在1964年推出的产品。它发展了1962年该公司首先在B 5000上体现的堆栈存储器的思想，成为了著名的堆栈式计算机。本章第八节介绍这台机器。

此外，在专用控制机方面，如AN/FSQ7和RW300，在小型机方面，如IBM 1620（主要用于科学计算）和1401（主要用于数据处理）都很有名。法国的多道程序计算机GAMMA 60和英国的堆栈式计算机KDF 9也有特色。但限于篇幅本书不一一介绍了。

日本从第二代开始大力发展计算机工业。虽然没有著名的机种出现，但它的工业基础和技术力量使其在第三代以后跃居世界第二位。与它相反是苏联，从第二代计算机以后就没有显著进展。当然原因很多，但这个领域的科研工作不面向生产和应用是其中的一个。英国计算机工业没有充分发展，尽管英国学者提出了很多有价值的思想，并在实验中取得效果，但最终还是被其它国家的计算机工业接受过去。欧洲国家的计算机工业都有不同程度的发展，发展中国家也开展了计算机的研制或使用。虽然从1964年IBM公司发表360系列以后，计算机进入了第三代，但第二代计算机在七十年代末仍有使用。

体系结构的发展是从提高运算和工作速度开始的。提出了很多改进四则运算的方法。当感到仅仅改进四则算法已不够时，人们从控制信息流方面提出改进措施。流水线和多部件并行是两种有代表性的措施。在存储器方面，为了解决存取速度与存储容量的矛盾，提出了分层控制和一级存储器思想。在输入输出设备方面，为了应付种类数量日益繁多而速度与主机的差距越来越大的外围设备，提出了通道和外围处理机等的独立控制。为了进一步提高计算机的可靠性，已从第一代力求减少元件而发展到采用错误检测码和错误校正码，用增加元件的办法实现用不可靠元件做可靠计算机的思想。使用方式对计算机体系结构产生影响。多道程序要求建立中断系统和存储保护。高级语言促进堆栈结构的应用。各式各样的应用领域对计算机体系结构都提出了要求。但所有这一切都是在单机结构上发展的。第二代计算机主要是单指令流单数据流的单机结构。

本章将沿着上述顺序进行介绍。

一。提高运算速度的方法

由于早期计算机主要用于科学计算，因此把提高运算速度，特别是四则运算速度做为提高整机工作速度的主要途径。在五十年代和六十年代，曾提出很多提高四则运算速度的方法。在七十年代以后，情况逐渐发生变化。人们的注意力已主要不放在改进算法提高运算速度方面，而放在信息流动的组织方面。另外，也提出了一些新的算法，这类算法在早期是无法实现的，而现在由于硬件成本的降低，已经可

以实现了。

本节介绍加法、乘法、除法和移位的几种提高速度的方法。

1．加法运算

两个二进制数的相加，首先使各个数位并行相加，但进位传送却要一位位地传递。加法的最后完成，实质上取决于进位传递的结束。因此，加快进位传送是提高加法速度的主要措施。

当然可以采用特殊元件提高进位传递速度，但本节主要介绍逻辑方法。这种方法有：保留进位、检测进位、小组进位、同时进位、条件和加法和金字塔式加法。在文献1中曾对除最后一种方法外的其余方法进行了综合比较。

保留进位加法：当两个数相加时，得到的半加和及进位分别存在两个寄存器内。当继续进行加法时，前次进位参与两个数的相加，又重新形成半加和及进位。只有在最后一次加法时，才进行完整的进位传送。这种方法在连续累加时有优越性，而且连续累加的数据越多，优点越显著。反之，如果不进行连续累加，则效果不大。

检测进位加法：增加一个进位检测线路，直接检测各位的进位。当每一位上都不存在进位时，就认为加法完成。这种方法并没有采用特别措施提高速度，只是由于加法进位传送时间不一，最长时间是进位传送经过每一位，而最短时间可能没有进位。平均进位传送经过的位数大约为总数的平方根。这种加法线路只适合于异步逻辑，同步逻辑总要保证最长进位的完成。

小组进位加法：这种方法也叫小组跳跃。将全部位数分成若干小

组。小组长度可以相等，也可能不等。在小组内部，进位是逐位传递（也称为行波式传递）。每组设一个就绪检测线路。当它判断出，两个数的半加和在该组内各个位都为1时，则允许将传递到这个组的进位，直接送到下一组去。同时进位还在组内传递，以决定各位的结果。这样一来，当字长一定时，组内位数与分成的组数之间就有一定的关系。若组内位数过多，延迟时间取决于组内的进位传递。若组内位数过少，延迟时间取决于组间进位传递。文献2中指出最佳分组办法应保持组数是组内位数的两倍。另外，经过分析认为：若分成不等长组，对一个字来讲，左右两端定为一位一组，然后朝内增加，每组增加一位，成为两边小、中间大的形式最好。如36位字长的数据，可分为11组，每组位数是：

1，2，3，4，5，6，5，4，3，2，1。

但不等长分组，结构异常复杂，不便于做成集成电路，现已不采用。

同时进位（图1）。将一个数据分成若干小组后，小组内部可以构成一个较为复杂的逻辑网络。当从前一组传来进位或组内某些位中形成进位时，它可以通过逻辑网络直接传递到每一位及小组出口。这就形成了小组内部同时进位。如果用这套逻辑网络控制组间进位传递，则构成组间同时进位。如果二者同时具备，就形成了全字同时进位。它被认为是传递速度最快的，但所用器材量也是较大的。

条件和加法：（图2）每一位都做出两种假定，低位有进位和低位无进位。第一步就根据这个假定每位都求出两组和及进位。第二步将每两位分为一组，并假定前一组对本组可能有进位或无进位，求出组内的

各位的和及本组向下一组的进位，也有两种可能性。第三步分成四位一一组，也照上述办法处理。由此类推，直到求出各位和为止。这种办法使用元件的数量多，但要求不高（只需两个输入端），而且有规律，适于使用集成电路实现。

金字塔式加法（图3）：它和条件和加法在组织上有类似之处，但不要求得到两组可能的值。它是首先逐位求出决定和及进位的必要参数，使得只要前一位的进位到来，就可以决定本位的和及进位。第二步是每两位分为一组，也是决定各种必要的参数，使得前一组进位到来后，就可以决定本组的各位和及向下一组的进位。第三步是将四位分成一组，也按上述办法处理。由此类推。从逻辑结构看，它相当于按2的幂数分组的同时进位结构。它使用的器材比条件和加法少，但控制逻辑较为复杂，规律性也较差。

2．乘法运算

乘法按照常规是移位累加。因此，减少累加次数是提高乘法速度的主要措施。现在采用的方法大体分为两类：移位累加方式和并行相乘方式。移位累加方式是早期提出的方法。乘数和部分积同时移位（左移或右移），当移出乘数为1时，将被乘数加到部分积上。提高速度的主要途径是减少加法次数。并行乘法则是采用大量的加法线路，直接用硬件实现累加。当时这种方法由于使用器材大多而不实际，但随着大规模集成电路的发展，它不仅可以实现，而且有发展的趋势。

移位累加方式中提高速度的办法有：保留进位乘，跳1跳0乘，多位乘法。

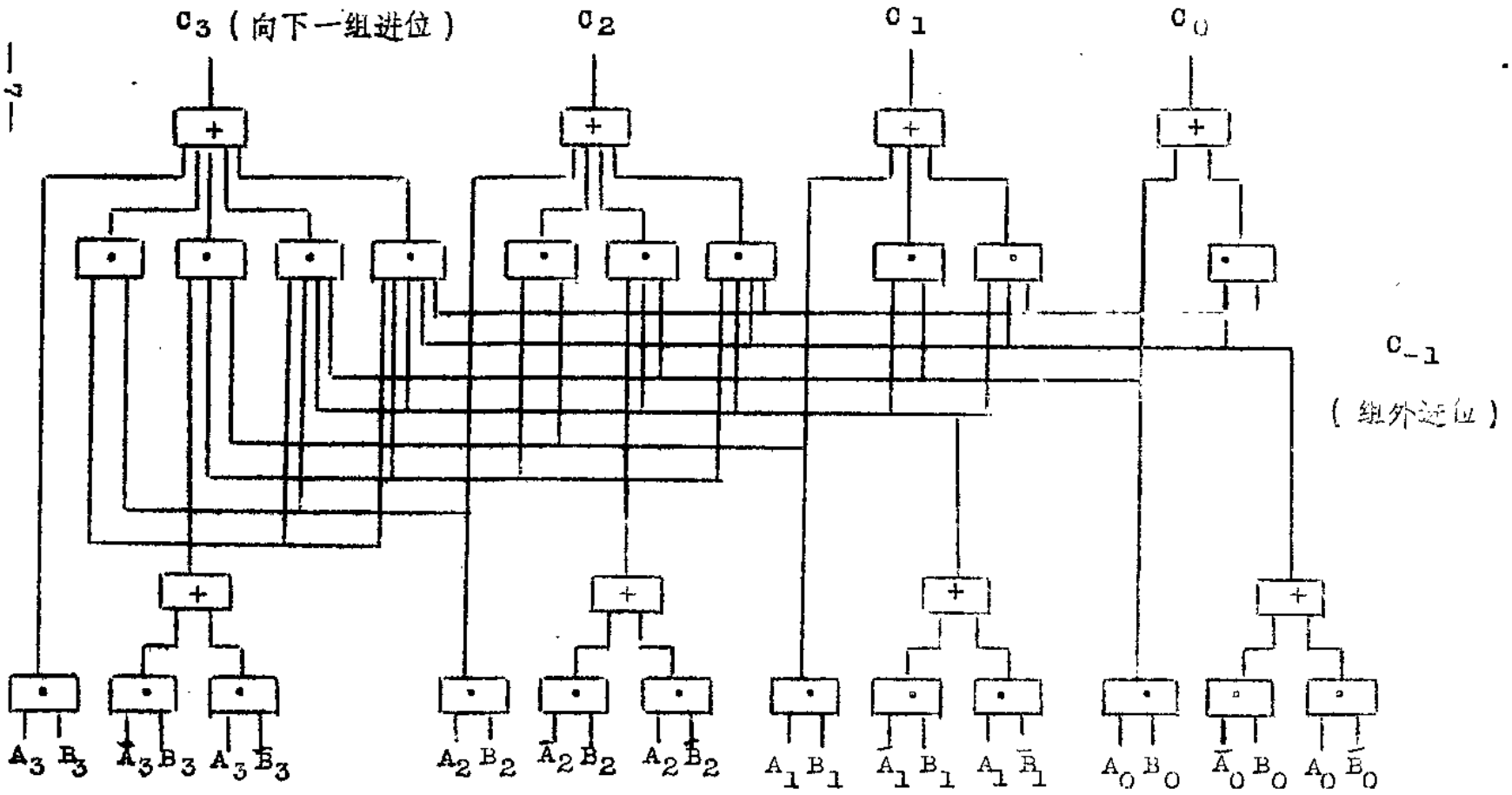
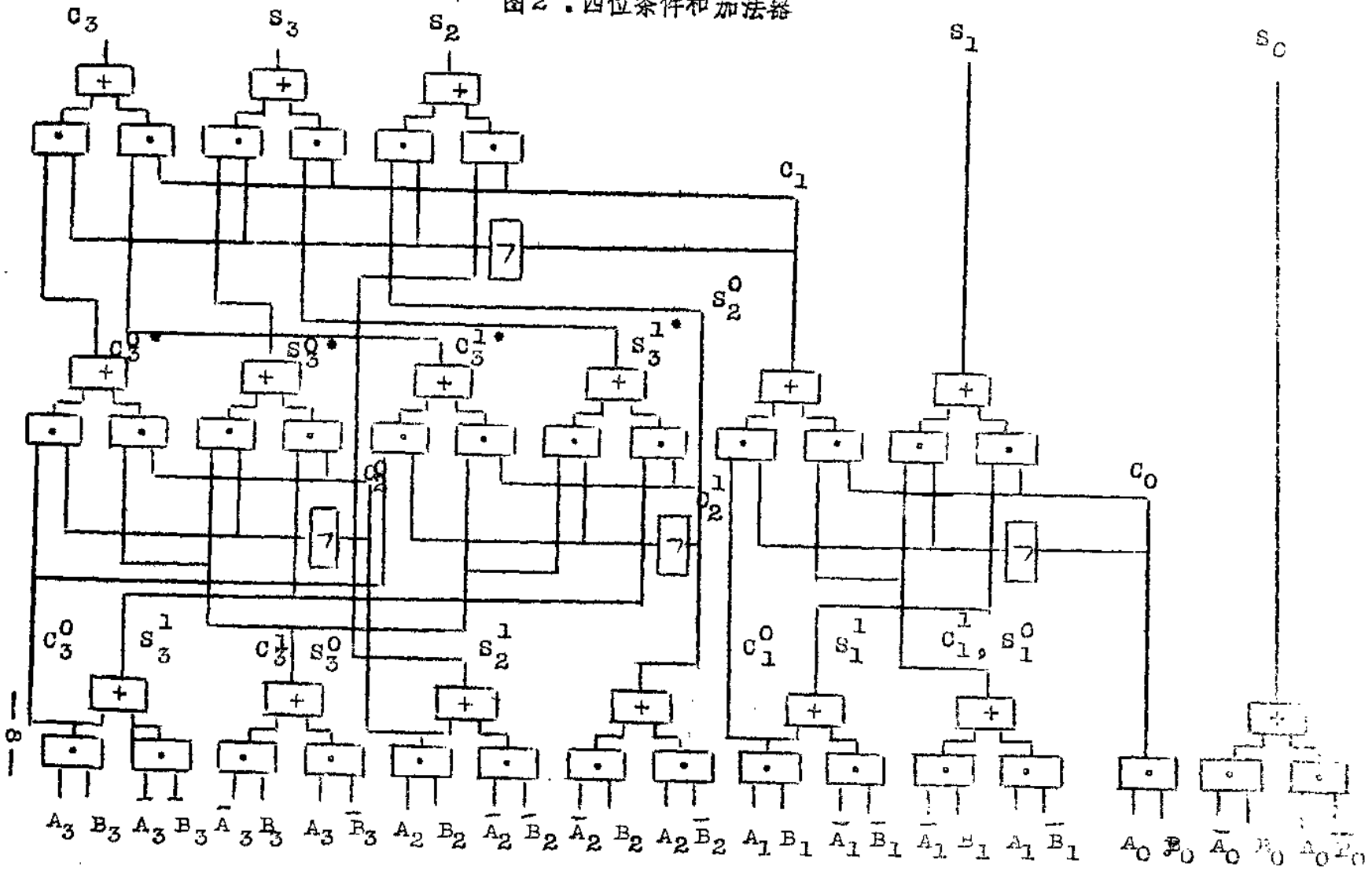

C3（向下一组进位）
C2
C1
C0
C-1
（组外进位）
A3 B3
A3 B̄3
Ā3 B3
A2 B2
A2 B̄2
Ā2 B2
A1 B1
A1 B̄1
Ā1 B1
A0 B0
A0 B̄0
Ā0 B0

图 1．四位小组同时进位

图2．四位条件和加法器


C_3
S_3
S_2
S_1
S_0
C_1
C_0
S_2^0
C_3^0
S_3^0
C_3^1
S_3^1
C_2^0
C_2^1
C_3^0
S_3^1
C_3^1
S_3^0
S_2^1
C_1^0
S_1^1
C_1^1, S_1^0
A_3 B_3 $\bar{A}_3$ B_3 A_3 $\bar{B}_3$ A_2 B_2 $\bar{A}_2$ $\bar{B}_2$ $\bar{A}_2$ B_2 A_2 $\bar{B}_2$ A_1 B_1 $\bar{A}_1$ $\bar{B}_1$ $\bar{A}_1$ B_1 A_1 $\bar{B}_1$ A_0 B_0 $\bar{A}_0$ B_0 A_0 $\bar{B}_0$

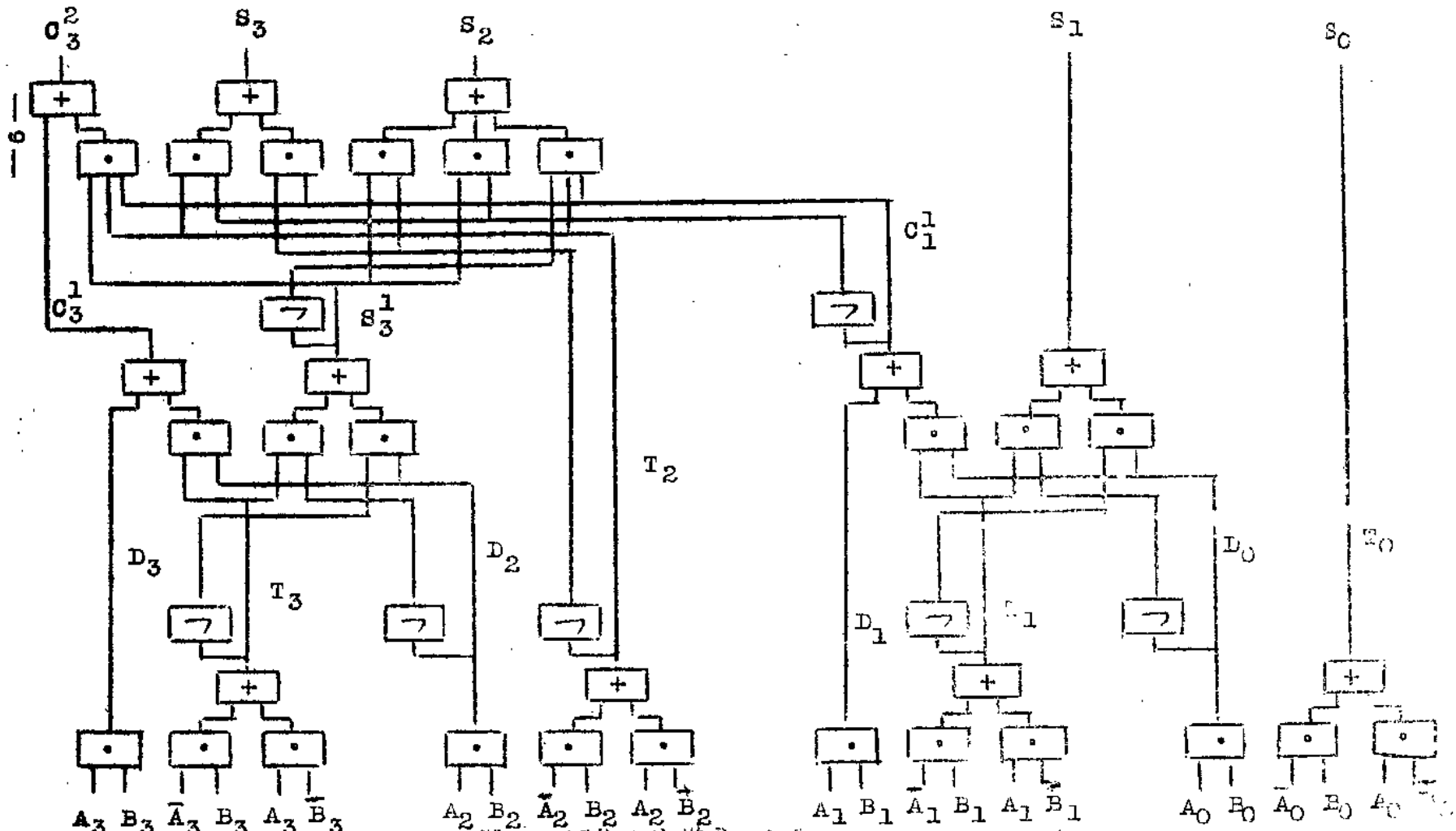


图 3 . 四位金字塔式加法器

保留进位乘：由于乘法是连续累加，前面介绍的保留进位加法完全适用于乘法。虽然没有减少加法次数，但由于取消了中间加法的进位传送，使加法时间缩短，从而提高了乘法速度。

跳1跳0乘：当乘数中出现连续个0时，可以直接移过不加被乘数到部分积上。当乘数中出现连续个1时，可以把这一部分看做是更高一位的1减去最低一位的1所得的结果。因此，可以在连续1的两端分别做一次加减法即可。具体方法是乘数和部分乘积右移。当乘数中遇到连续0或连续1时，则直接移过。当出现1、0交错时，要按下列规则进行加、减：第一，移过0而停于1时，若1后为0则加被乘数，若1后为1则减被乘数；第二，移过1而停于0时，若0后为1则减被乘数，若0后为0则加被乘数。按这两项规则加减后继续右移。

多位乘法：可以将乘数每次移过n位而进行一次累加，由此减少移加次数。多位乘法与保留进位乘结合起来是目前常用的方法，下面具体介绍几个例子：

两位乘法：乘数每次右移两位。若移出的乘数为0、1（即高位为0），则根据低位为0或1，决定不加或加一倍被乘数。若移出的乘数为2、3（即高位为1），则根据低位为0或1，决定减两倍或减一倍被乘数，并将高位的1保留，与下次移出的乘数混合控制，使其相当于将下次移出的乘数加1。具体控制规则如表1所示。

四位乘法：乘数每次右移四位。按照四位乘数的各种组合，对乘数的一倍、两倍、四倍、八倍、十六倍正负值进行最多三次连加，形

成部分积。乘法规则如表2所示。所说求方式实现它的过程请参看本章下一节。

十二位乘法：它是一种由简单多位构成复杂多位的实例。乘数每次右移12位。移出的乘数分为三组，四位一组。每组按上述四位乘法规则进行相加。三组并行计算，所得结果再叠加一起获得最后部分积。但是完全按照上述规则，四位乘法最多要连加三次。为了将它减为二次，可按照两位乘法的办法，将每组的高位与下一组重叠译码，使下一组相当于低位加1。乘法规则表列于表3。采用多层保留进位加法器可以实现这一规则。图4是FACOM 230-75计算机中，采用这一规则实现的过程。其中F寄存器中存有各种控制开门信号，并分为六组。相加时，每组只开一个门。

并行乘法：假如用大量的加法器，完全仿照乘法过程排列成一个阵列，则整个叠加过程就是流过这个加法阵列的过程，速度会大大提高。

表1．两位乘法规则

乘数		前次乘数高位	操作*	乘数		前次乘数高位	操作
0	0	0	—	1	0	0	$-2d$
0	0	1	$+d$	1	0	1	$-d$
0	1	0	$+d$	1	1	0	$-d$
0	1	1	$+2d$	1	1	1	—

*d为被乘数

表2．四位乘法规则

乘数	操作	乘数	操作
0 0 0 0	—	1 0 0 0	$+8d$
0 0 0 1	$+d$	1 0 0 1	$8d+d$
0 0 1 0	$+2d$	1 0 1 0	$8d+2d$
0 0 1 1	$4d-d$	1 0 1 1	$16d-4d-d$
0 1 0 0	$+4d$	1 1 0 0	$16d-4d$
0 1 0 1	$4d+d$	1 1 0 1	$16d-4d+d$
0 1 1 0	$8d-2d$	1 1 1 0	$16d-2d$
0 1 1 1	$8d-d$	1 1 1 1	$16d-d$

图5示出一个5位×5位的并行乘法器。这种乘法器要求两个数应为无符号数。如果是有符号数，而且用反码或补码表示时，则在乘之前，应变为原码。乘得的积还要恢复为反码或补码。这增加了乘法时间。为此，提出了很多直接用补码进行并行乘法的方案，可参看文献〔4〕。

3．除法运算

除法运算与乘法运算有类似的过程。它将除数从被除数中减去，根据余数的正负，决定商值，并决定下一次是从左移后的余数中减去除数，还是加上除数。然后再重复这个加减——判断——移位的过程。因为通常每求一个商就要重复这一过程，所以提高除法速度的办法主要是在重复这一过程时求得更多位的商。但目前除法方法已发展为三

类：一类是在重复这一过程时求得更多位的商，如多位除法，一类是用乘法代替除法，如收敛除法；还有一类是阵列除法。

多位除法：实际使用过二位除法。首先求出除数的一倍、二倍和三倍值，然后用被除数或余数同时去减它们，根据得到的结果决定商值。显然，这种除法比二位乘法要复杂得多。

收敛除法：它的原理是假定

$$N / D = Q$$

从而有

$$\frac{N}{D} = \frac{N}{D} \times \frac{R_0}{R_0} \times \frac{R_1}{R_1} \times \frac{R_2}{R_2} \times \cdots\cdots \times \frac{R_n}{R_n} \quad .$$

若将 n 取值很大，并使得

$$D \times R_0 \times R_1 \times R_2 \times R_3 \cdots\cdots R_n = 1,$$

则可得 $N \times R_0 \times R_1 \times R_2 \times R_3 \cdots\cdots \times R_n = Q$

问题是如何求得 R_0、R_1、R_2、……R_n。若 D 和 N 都是规格化小数，则可以假定

$$1 - D = \delta$$

并选择第一个因子 $R_0 = 1+\delta$，然后选择 $R = 1+\delta^2$ …… $R_i = 1+\delta^i$ 等。

在 FACOM 230-75 中，这个算法的实现过程是：根据除数 D 的前 9 位，确定能迅速收敛的 R_0 的值，做成表格。每次计算时直接取入，

表 3 . 四位乘法规则（另一种形式）

乘数 *	乘法门控制信号 -2^3	$+2^3$	-2^2	$+2^2$	$+2^1$	-2^0	$+2^0$	操作 **	乘数 *	乘法门控制信号 -2^3	$+2^3$	-2^2	$+2^2$	$+2^1$	-2^0	$+2^0$	操作 **
0 0 0 0 0	0	0	0	0	0	0	0	—	1 0 0 0 0	1	0	0	0	0	0	0	$(16d)-8d$
0 0 0 0 1	0	0	0	0	0	0	1	$+d$	1 0 0 0 1	1	0	0	0	0	0	1	$(16d)-8d+d$
0 0 0 1 0	0	0	0	0	0	0	1	$+d$	1 0 0 1 0	1	0	0	0	0	0	1	$(16d)-8d+d$
0 0 0 1 1	0	0	0	0	1	0	0	$+2d$	1 0 0 1 1	1	0	0	0	1	0	0	$(16d)-8d+2d$
0 0 1 0 0	0	0	0	0	1	0	0	$+2d$	1 0 1 0 0	1	0	0	0	1	0	0	$(16d)-8d+2d$
0 0 1 0 1	0	0	0	1	0	1	0	$4d$ $-d$	1 0 1 0 1	0	0	1	0	0	1	0	$(16d)-4d-d$
0 0 1 1 0	0	0	0	1	0	1	0	$4d$ $-d$	1 0 1 1 0	0	0	1	0	0	1	0	$(16d)-4d-d$
0 0 1 1 1	0	0	0	1	0	0	0	$+4d$	1 0 1 1 1	0	0	1	0	0	0	0	$(16d)-4d$
0 1 0 0 0	0	0	0	1	0	0	0	$+4d$	1 1 0 0 0	0	0	1	0	0	0	0	$(16d)-4d$
0 1 0 0 1	0	0	0	1	0	0	1	$4d$ $+d$	1 1 0 0 1	0	0	1	0	0	0	1	$(16d)-4d+d$
0 1 0 1 0	0	0	0	1	0	0	1	$4d$ $+d$	1 1 0 1 0	0	0	1	0	0	0	1	$(16d)-4d+d$
0 1 0 1 1	0	0	0	1	1	0	0	$4d+2d$	1 1 0 1 1	0	0	1	0	1	0	0	$(16d)-4d+2d$
0 1 1 0 0	0	0	0	1	1	0	0	$4d+2d$	1 1 1 0 0	0	0	1	0	1	0	0	$(16d)-4d+2d$
0 1 1 0 1	0	1	0	0	0	1	0	$8d$ $-d$	1 1 1 0 1	0	0	0	0	0	1	0	$(16d)-d$
0 1 1 1 0	0	1	0	0	0	1	0	$8d$ $-d$	1 1 1 1 0	0	0	0	0	0	1	0	$(16d)-d$
0 1 1 1 1	0	1	0	0	0	0	0	$8d$	1 1 1 1 1	0	0	0	0	0	0	0	$(16d)$

* 最后一位为前次乘数高位

** d 为被乘数