第2卷 第4期 电 子 学 通 讯 Vol. 2, No. 4
1980 年 10 月 JOURNAL OF ELECTRONICS Oct., 1980

对称开关逻辑的三值胞腔阵列*

郑 启 伦
(华南工学院)

提 要

本文以三元素取值为 $\{\bar{1}, 0, 1\}$的对称开关逻辑为基础,提出一种可用于实现任何组合功能的规律和半规律的阵列结构,讨论了这种结构的设计原理,介绍了若干阵列的布局方式,并把 Agrawal 最近设计的四位二进制高速乘法器阵列修改为本文定义的三进制高速乘法器阵列. 最后,通过与二进制阵列的比较,证明了对称开关逻辑的三值胞腔阵列的研究价值.

一、引 言

随着大规模集成技术发展的需要,逻辑设计正转向电路标准化、高存储密度、简化布线、容易设计和便于检测等问题方面. 如果采用一种由相同的基本电路(细胞)以规律或半规律排列方式组成的所谓胞腔阵列,上述问题大多数可以得到解决[1-3]. 关于这方面的研究过去曾发表过许多文献,但大多数是以二值逻辑为基础的,它们往往由于使用太多的细胞而令人失望.

近年,多值逻辑已经成为一门专门研究的学科,由于它具有远比二值逻辑更丰富的结构,同时在简化布线、提高存储密度、提高速度以及微程序的实现等方面的优点很突出,因此已引起人们的重视[4-6]. 但关于多值逻辑在胞腔阵列方面的研究文章仍然较少.

在目前条件下,实现以$\{\bar{1}, 0, 1\}$三元素取值的对称开关逻辑是有意义的,因为它既比用多电平(电流)模拟的高基数多值逻辑更易于实现,又比通常的二进制系统具有更多的优点[6-8]. 如果采用本文定义的三值细胞,则有可能弥补二值胞腔阵列必须使用太多细胞的缺陷,有利于大规模和超大规模集成化,而且这种规律或半规律结构的实现,将为未来实现多层器件的集成技术提供条件,使之成为一种"超"超大规模集成技术的新概念[9].

二、基本细胞和胞腔阵列

满足下述定义(1)的三值三方向开关单元称为三值细胞(图 1),即

定义 (1)

* 1979 年 9 月 14 日收到.

$$f=\begin{cases}I_x & \text{当 } C=\bar{1}.\\ I_y & \text{当 } C=0.\\ I_z & \text{当 } C=1.\end{cases}$$

其中 I_x、I_y 和 I_z 为三个水平方向输入，C 为垂直方向控制输入，f 为细胞的三个水平方向输出，且 I_x、I_y、I_z 和 $C\in L=\{\bar{1},0,1\}$.

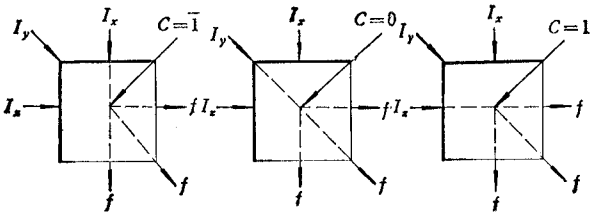


图 1 基本的三值细胞

若将三元素取值，且令 $\bar{1}<0<1$，可定义如下逻辑运算：

定义 (2)

$$\bigvee_{i=0}^{n} C_i=(C_0,C_1,\cdots,C_n)\max.$$

定义 (3)

$$\bigwedge_{i=0}^{n} C_i=(C_0,C_1,\cdots,C_n)\min.$$

定义 (4)

$$C_i^{K_i}=\begin{cases}1 & \text{当 } C_i=K_i.\\ \bar{1} & \text{当 } C_i\neq K_i.\end{cases}$$

其中 C_i 为某第 i 个细胞的控制输入变元；K_i 为其指数，且 $K_i\in L$. 定义 (2) 称为逻辑“或”，以算子“$\vee$”表示. 定义(3)称为逻辑“与”以算子“$\cdot$”表示. 定义(4)则称为控制变元的矢量函数.

由此得任意一个三值细胞的特征函数为：

$$f_i=C_i^{\bar{1}}\cdot I_x\vee C_i^{0}\cdot I_y\vee C_i^{1}\cdot I_z. \tag{1}$$

由式 (1) 知：一个三值细胞可以采用已发表的各种三值开关电路并按图 2 的框图联接而实现[10-12]，或者采用如图 3 所示的一种具有回折集电极结构的 I^2L 电路来实现. 这种电路是作者根据文献[13]图 8(b) 作进一步修改而得到的. 因为所有横向管（恒流源）采用了相同的射集电流 (20μA)，所以电路的设计和制作的困难减少了.

所研究的三值胞腔阵列由相同的三值细胞按二维点阵方式排列. 结构形式类似于 Akers[3] 和 Kukreja[1] 的二值胞腔阵列. 不同的是每一个三值细胞在二值细胞的基础上增加一个输入和输出端，且每一个细胞可传输一个三值信息. 整个模块中细胞之间仍然是一种无交叉布线的可密集的排列方式，有利于集成化（如图 4).

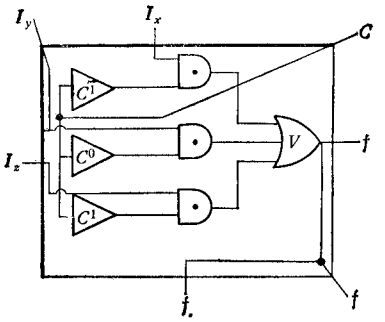


图 2　三值细胞的实现

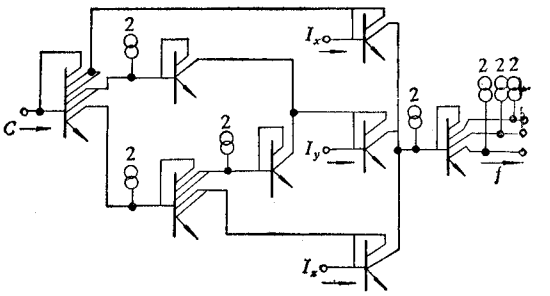


图 3　I^2L 的三值细胞电路

定义 (5)

一个具有 P^2 个三值细胞构成的二维阵列，其中某一点细胞 $i = i(x, z)$，若将 x 和 z 视为二维坐标值，则

$$i(x, z) = x + z \times P. \qquad (2)$$

如图 4 可视为一个 $P = 5$ 的阵列图，其中若某 i 点细胞的坐标值 $x = 2$、$z = 2$，则记为 $i = 2 + 2 \times 5 = 12$.

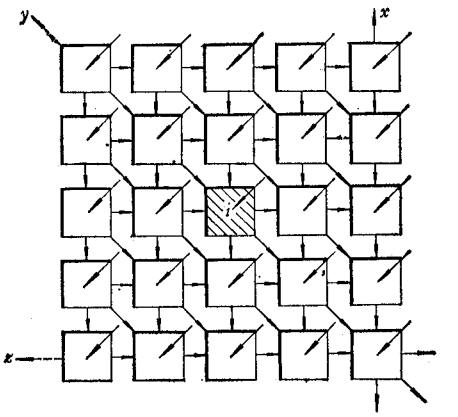


图 4　三值胞腔阵列 ($P = 5$)

根据定义(1)和图 4 知：每一个瞬间，任何一个三值细胞的输出端只能与其中一个输入端相接．在阵列中此输入端便是相邻细胞的输出．所以同一瞬间任何一个细胞的输入端只存在一个相邻细胞，且称它为该细胞的直接前驱.

性质 (1)

在 P^2 个细胞组成的阵列中，若某细胞 j 为细胞 i 的直接前驱，它们必然满足下述关系：

$$j = \begin{cases} i+1 & \text{当 } C_i = \bar{1},\ \text{且 } i \neq m \times P + P - 1. \\ i+P+1 & \text{当 } C_i = 0,\ \text{且 } i \neq m \times P + P - 1,\ i \neq m + P^2 - P. \\ i+P & \text{当 } C_i = 1,\ \text{且 } i \neq m + P^2 - P. \end{cases} \qquad (3)$$

其中 $m = 0, 1, 2, \cdots, P-1$，不满足上述条件时 $j = \phi$ (空集).

因此，三值细胞特征函数式(1)的另一种表达式为：

$$f_i = \{C_i^{K_i} \cdot f_j \mid C_i = K_i\} \qquad (4)$$

若知道某 i 点细胞及其逐级前驱细胞的控制值，便可求出相应的直接前驱细胞的位置(但 $j \neq \phi$).

例如：在 $P = 5$ 的阵列中(图 4)，已知 $i = 0$ 且 $C_0 = 0$，其逐级前驱细胞的控制值为 $C_{j1} = \bar{1}$、$C_{j2} = 1$、$C_{j3} = 1$、$C_{j4} = \bar{1}$ 和 $C_{j5} = 0$，试求 $f_0 = ?$

[解]由式(4)知：当 $C_i = K_i$ 时有：$f_0 = C_0^0 \cdot C_{j1}^{\bar{1}} \cdot C_{j2}^1 \cdot C_{j3}^1 \cdot C_{j4}^{\bar{1}} \cdot C_{j5}^0 \cdot f_?$

根据式(3)计算下角标值得：$f_0 = C_0^0 \cdot C_6^{\bar{1}} \cdot C_7^1 \cdot C_{12}^1 \cdot C_{17}^{\bar{1}} \cdot C_{18}^0 \cdot f_{24}$.

因为当 $C_i = K_i$ 时，所有矢量函数为"1"，故 $f_0 = f_{24}$.

因为 $i = 24$ 不满足式(3)的条件，所以它不再存在直接前驱，实际上只能是阵列边沿的一个输入 I_{24}，所以 $f_0 = I_{24}$（见图5）.

显然，此时函数中的所有矢量函数之积是有序的，且可以由它们的上角标 K_i 的序列来表示，其序列记为 $e = \langle 0\ \bar{1}\ 1\ 1\ \bar{1}\ 0\rangle$，当序列变更时，由它所描述的路径也将随之变化.

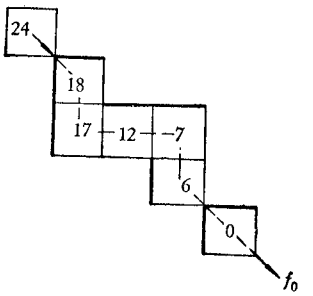


图5 由控制值所决定的一个路径

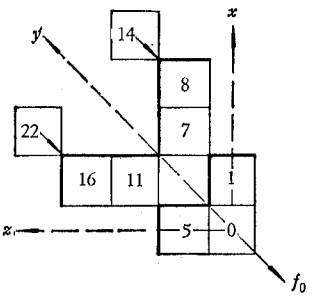


图6 $f_0 = I_{14}$ 和 $f_0 = I_{22}$ 的两个路径

性质 (2)

在 P^2 个细胞组成的阵列中，相同个数 K_i 值的序列，无论它们的排列顺序怎样，只要 $j \neq \phi$，由它们组成的所有矢量函数之积，均表达了某细胞的输出是来自于另一个细胞的所有可能路径.

如上例中 K_i 序列 $e = \langle 0\ \bar{1}\ 1\ 1\ \bar{1}\ 0\rangle$ 可改变为$\langle \bar{1}\bar{1}0011\rangle$、$\langle \bar{1}\bar{1}0101\rangle$、$\langle \bar{1}\bar{1}0110\rangle$、……等，根据式(4)和性质(1)的下角标值计算，得到如下各项积之"或"：

$$\begin{aligned} f_0 = {} & C_0^{\bar{1}} \cdot C_1^{\bar{1}} \cdot C_2^0 \cdot C_8^0 \cdot C_{14}^1 \cdot C_{19}^1 \cdot f_{24} \\ \vee\ & C_0^{\bar{1}} \cdot C_1^{\bar{1}} \cdot C_2^0 \cdot C_8^1 \cdot C_{13}^0 \cdot C_{19}^1 \cdot f_{24} \\ \vee\ & C_0^{\bar{1}} \cdot C_1^{\bar{1}} \cdot C_2^0 \cdot C_8^1 \cdot C_{13}^1 \cdot C_{18}^0 \cdot f_{24} \\ & \vdots \\ \vee\ & C_0^1 \cdot C_5^1 \cdot C_{10}^0 \cdot C_{16}^0 \cdot C_{22}^{\bar{1}} \cdot C_{23}^{\bar{1}} \cdot f_{24} \end{aligned}$$

因为 $f_{24} = I_{24}$，所以上式中每一积项均表达了 $f_0 = I_{24}$ 的一种可能路径. 那么所有上角标 K_i 序列的集合 E，只要其中 $j \neq \phi$，便是构成 $f_0 = I_{24}$ 的路径集合，且记为：

$$E_{24\to 0} = \{\langle \bar{1}\bar{1}0011\rangle, \langle \bar{1}\bar{1}0101\rangle, \langle \bar{1}\bar{1}0110\rangle, \cdots, \langle 1100\bar{1}\bar{1}\rangle\}$$

根据性质(2)，如果为实现 $f_0 = I_{24}$，则可在上述序列中选择一个合适的路径，分配恰当的控制变元.

性质 (3)

在 P^2 个细胞组成的阵列中，已知某 $f_i = f_n$ 的 K_i 序列的集合 $E_{n\to i}$，那么 $\overline{E_{n\to i}}$（即 $1\Rightarrow\bar{1}$，$0\Rightarrow 0$，$\bar{1}\Rightarrow 1$）必然是 $f_i = I_n$ 对于以 y 轴线为镜的映射 $f_{i'} = f_{n'}$ 的 K_i 序列的集合 $E_{n'\to i'}$，如果 $i = i(x_1, z_1)$，$n = n(x_2, z_2)$，则 $i' = i'(z_1, x_1)$，$n' = n'(z_2, x_2)$.

例如：在 $P = 5$ 的阵列中(图4)，选择 $f_0 = I_{14}$ 的一个路径序列为 $e_{14\to 0} = \langle \bar{1}0\bar{1}0\rangle$，

试求与之对称的另一路径.

[解] 由式(4)知：当 $C_i = K_i$ 时，有 $f_0 = C_0^{\bar{1}} \cdot C_1^0 \cdot C_7^{\bar{1}} \cdot C_8^0 \cdot I_{14}$ 当 $\overline{e_{14\to 0}} = \langle 1010 \rangle$，则有 $f_0 = C_0^1 \cdot C_5^0 \cdot C_{11}^1 \cdot C_{16}^0 \cdot I_{22}$

即 $\overline{e_{14\to 0}} = e_{22\to 0}$. 根据定义(5)下角标值的计算：当 $i(x_1, z_1) = 0$，则 $x_1 = z_1 = 0$；当 $n = n(x_2, z_2) = 14$，则 $x_2 = 4$，$z_2 = 2$. 因 $i = i'(0, 0) = 0$，$n'(2, 4) = 22$. 上述计算符合性质(3). 图 6 便是 $f_0 = I_{14}$ 和 $f_0 = I_{22}$ 的两个路径. 从图中可以看出它们是以 y 轴线为镜的一对映射.

性质(3)阐明了结构的对称性，利用它对于某些较复杂的阵列设计，可达到简化设计步骤的目的，因为只要设计好阵列的一边，便可容易地得到同样正确的另一边.

三、常用阵列的设计

许多复杂的多变元组合逻辑系统，常常可以分解为若干个较简单的模块. 这些模块将占据阵列的全部或局部面积. 每个模块内具有相同的互连，而模块与模块之间则采取规律或半规律的连接方式，这样任何复杂的功能均可以实现. 本节以讨论某些常用模块设计为主，对于模块与模块之间的组合在下一节讨论.

图 7 给出了对应于二变元 A 和 B 的各种真值表形式及其胞腔模块，其中①，②，……，⑤表示各种输出状态，它们可以是某函数也可以是某逻辑常数，并且从模块的不同边沿输入.

以图 7(b) 的设计为例：由真值表列出其相应的函数为：

$$f = A^{\bar{1}} \cdot B^{\bar{1}} \cdot ① \vee A^{\bar{1}} \cdot B^0 \cdot ② \vee A^{\bar{1}} \cdot B^1 \cdot ③$$
$$\vee A^0 \cdot B^{\bar{1}} \cdot ③ \vee A^0 \cdot B^0 \cdot ③ \vee A^0 \cdot B^1 \cdot ③$$
$$\vee A^1 \cdot B^{\bar{1}} \cdot ③ \vee A^1 \cdot B^0 \cdot ④ \vee A^1 \cdot B^1 \cdot ⑤.$$

化简得：

$$f = A^{\bar{1}} \cdot B^{\bar{1}} \cdot ① \vee A^{\bar{1}} \cdot B^0 \cdot ② \vee A^{\bar{1}} \cdot B^1 \cdot ③$$
$$\vee A^0 \cdot ③ \vee A^1 \cdot B^{\bar{1}} \cdot ③ \vee A^1 \cdot B^0 \cdot ④ \vee A^1 \cdot B^1 \cdot ⑤.$$

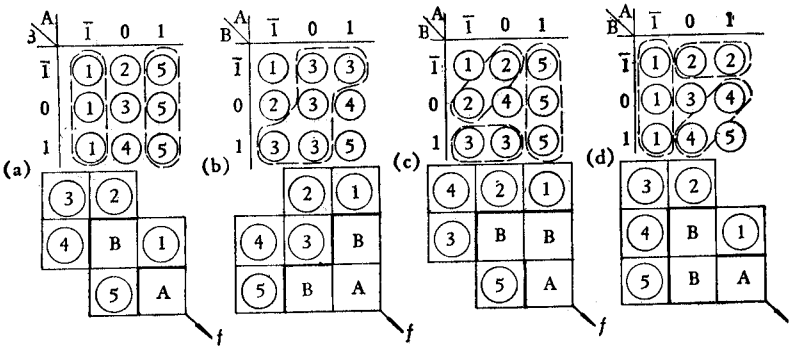


图 7　若干个真值表形式及其相应模块

若将式中每一积项指定为相应的一个路径，并假设 $P = 3$，输出取自于 $i = 0$ 的细胞，则

有

$$f_0 = A_0^{\bar{1}} \cdot B_1^{\bar{1}} \cdot f_2 \vee A_0^{\bar{1}} \cdot B_1^{0} \cdot f_5 \vee A_0^{\bar{1}} \cdot B_1^{1} \cdot f_4$$
$$\vee A_0^{0} \cdot f_4 \vee A_0^{1} \cdot B_3^{\bar{1}} \cdot f_4 \vee A_0^{1} \cdot B_3^{0} \cdot f_7 \vee A_0^{1} \cdot B_3^{1} \cdot f_6.$$

显然，当 $f_2 =$ ①，$f_4 =$ ③，$f_5 =$ ②，$f_6 =$ ⑤ 以及 $f_7 =$ ④ 时，则符合设计的要求(见图 7 (b)的布向).

如果在某些情况下，根据给定的函数选择的路径发生矛盾，可根据上节的原理作相应的修改. 关于这个问题将在乘法器设计中举例.

下面介绍几种常用的阵列供读者参考:

1. “或”和“与”阵列

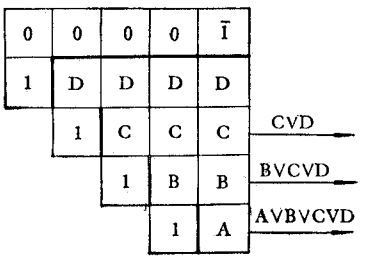
由定义(2)和定义(3)知: 某二变元 A 和 B 的三值逻辑“或”和“与”的真值表如表 1 和表 2 所示，显然它们相应于图 7(c) 和图7 (d) 的两个真值表形式，因此可以采用相应的两种模块实现它们. 图 8 是四变元的 “或”和“与” 阵列，它是图 7(c) 和图 7(d) 两个模块的扩展. n 变元的“或”和“与”阵列可依此规律构成.

表 1. “或” 的真值表

B \ A	$\bar{1}$	0	1
$\bar{1}$	$\bar{1}$	0	1
0	0	0	1
1	1	1	1

表 2. “与”的真值表

B \ A	$\bar{1}$	0	1
$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
0	$\bar{1}$	0	0
1	$\bar{1}$	0	1



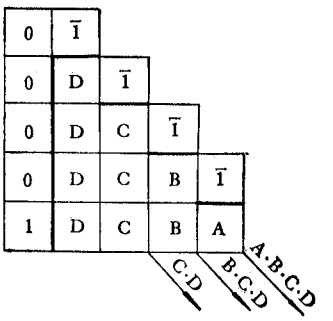


图 8 可扩展的“或”和“与”阵列

2. 三值系列脉冲发生器

一种采用两位二进制计数器和三值细胞相结合的三值系列脉冲发生器如图 9 所示，当模拟 $\bar{1}$ 和 1 的脉冲进入计数器后，计数器输出端 AB 按 $\bar{1}\bar{1}$, $\bar{1}1$, $1\bar{1}$ 和 11 四种状态变化. 当它们被馈送到相应的细胞后，模块将输出相应的三值系列脉冲.

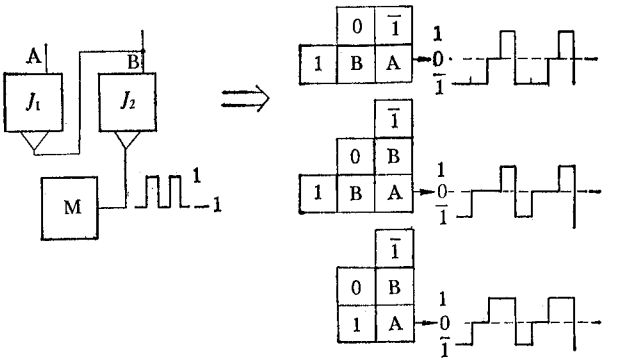


图 9　三值系列脉冲发生器

3. 译码器阵列

图 10(a) 是一个三位的译码器阵列. 它假设译码输出为 1,非译码输出为 0. 对应于一变元 A 的三态可选三位输出端. 图 10(b) 是图 10(a) 的扩展. 它对应于二变元 A 和 B 的九种组合可选九位输出端.

4. 通用逻辑阵列

n 变元的通用逻辑阵列可用以实现 n 变元三值逻辑的任何函数,也可视为一个 3^n 路开关. 每一路径允许一股三值信息流到达阵列的输出端. 图 11 便是 $n=1$—3 变元的通用逻辑阵列. 它们分别允许 3—27 个边沿输入,并且具有以 y 轴线为中心的对称性(见性质(3)),只要我们设计了对称的一边,便可以轻易地设计与之对称的另一边. 根据设计结果认为: n 变元三值逻辑通用阵列需要细胞的总数目为:

$$\theta = \sum_{i=0}^{i=(3^n-1)/2} i \tag{5}$$

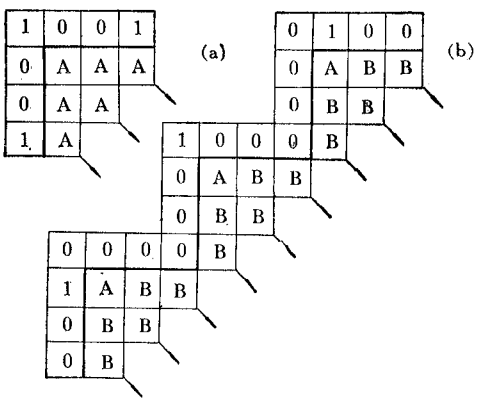


图 10　译码器阵列

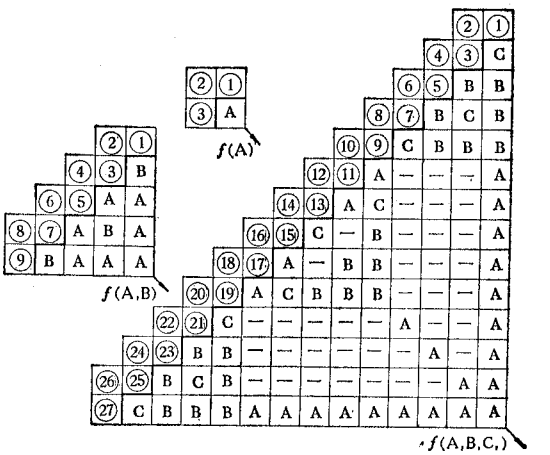


图 11　通用逻辑阵列 ($n=1$—3)

四、特殊阵列的设计

(一) 一位三进制乘法器和全加器阵列

表 3 列出了四位 "trits" 数码与十进制数的对应关系. 三进制数 $\bar{1}\bar{1}\bar{1}\bar{1}$—1111 对应于十进制数为−40—+40. 正数和负数是以 "0" 为中心的互补关系. 根据这个对称编码法,表 4 分别列出了一位三进制乘法器和全加器运算的真值表,其中 a 和 b 为相乘或相加变数, C^*为进位输入, m 为积, s 为和,而 c 为进位输出.

这种编码方式最大的优点是对正负数的运算非常方便,无须符号位及符号处理,例如:

$$\begin{array}{rl} & A = 010\bar{1} \quad (8) \\ +) & B = 0\bar{1}11 \quad (-5) \\ \hline & S = 0010 \quad (3) \end{array}$$

$$\begin{array}{rl} & A = 0\ \bar{1}\ 0\ 1 \quad (-8) \\ \times) & B = 0\ \bar{1}\ 1\ 1 \quad (-5) \\ \hline & \quad 0\ \bar{1}\ 0\ 1 \\ & 0\ \bar{1}\ 0\ 1 \\ & 0\ 1\ 0\ \bar{1} \\ & 0\ 0\ 0\ 0 \\ \hline & M = 0\ 0\ 0\ 1\ 1\ 1\ 1 \quad (40) \end{array}$$

表 3. 编码方法

1	1	1	1	40
1	1	1	0	39
⋮	⋮	⋮	⋮	⋮
0	1	$\bar{1}$	$\bar{1}$	5
0	0	1	1	4
0	0	1	0	3
0	0	1	$\bar{1}$	2
0	0	0	1	1
0	0	0	0	0
0	0	0	$\bar{1}$	−1
0	0	$\bar{1}$	1	−2
0	0	$\bar{1}$	0	−3
0	0	$\bar{1}$	$\bar{1}$	−4
0	$\bar{1}$	1	1	−5
⋮	⋮	⋮	⋮	⋮
$\bar{1}$	$\bar{1}$	$\bar{1}$	0	−39
$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	−40

表 4. 乘法器和加法器真值表

m \ b \ a	$\bar{1}$	0	1
$\bar{1}$	1	0	$\bar{1}$
0	0	0	0
1	$\bar{1}$	0	1

s \ b \ a \ c^*	$\bar{1}$			0			1		
	$\bar{1}$	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1
$\bar{1}$	0	1	$\bar{1}$	1	$\bar{1}$	0	$\bar{1}$	0	1
0	1	$\bar{1}$	0	$\bar{1}$	0	1	0	1	$\bar{1}$
1	$\bar{1}$	0	1	0	1	$\bar{1}$	1	$\bar{1}$	0

c \ b \ a \ c^*	$\bar{1}$			0			1		
	$\bar{1}$	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1
$\bar{1}$	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	0	0	0	0	0
0	$\bar{1}$	0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	0	1	1

根据表 4 可以列出一位乘法器的逻辑函数为：

$$m = a^{\bar{1}} \cdot b^{\bar{1}} \cdot 1 \vee a^{\bar{1}} \cdot b^{0} \cdot 0 \vee a^{\bar{1}} \cdot b^{1} \cdot \bar{1}$$
$$\vee a^{0} \cdot b^{\bar{1}} \cdot 0 \vee a^{0} \cdot b^{0} \cdot 0 \vee a^{0} \cdot b^{1} \cdot 0$$
$$\vee a^{1} \cdot b^{\bar{1}} \cdot \bar{1} \vee a^{1} \cdot b^{0} \cdot 0 \vee a^{1} \cdot b^{1} \cdot 1$$

(注意：作为边沿输入的逻辑常数不能按定义(1)和(2)作最大值或最小值处理)上式可化简为：

$$m = a^{\bar{1}} \cdot b^{\bar{1}} \cdot 1 \vee a^{\bar{1}} \cdot b^{0} \cdot 0 \vee a^{\bar{1}} \cdot b^{1} \cdot \bar{1}$$
$$\vee a^{0} \cdot 0 \vee a^{1} \cdot b.$$

若将每一积项视为一个路径，假设用一个 $P = 4$ 的阵列去实现 m 运算，输出取自于第 0 点细胞，那么：

$$f_0 = a_0^{\bar{1}} \cdot b_1^{\bar{1}} \cdot 1_2 \vee a_0^{\bar{1}} \cdot b_1^{0} \cdot 0_6 \vee a_0^{\bar{1}} \cdot b_1^{1} \cdot \bar{1}_5$$
$$\vee a_0^{0} \cdot 0_5 \vee a_0^{1} \cdot b_4.$$

显然，路径 $a_0^{\bar{1}} \cdot b_1^{1} \cdot \bar{1}_5$ 和 $a_0^{0} \cdot 0_5$ 发生重叠，这时可根据 $a \cdot a \cdot \cdots \cdot a = a$ 的关系，令 $a^{\bar{1}} \cdot a^{\bar{1}} = a^{\bar{1}}$，则有

$$f_0 = a_0^{\bar{1}} \cdot a_1^{\bar{1}} \cdot b_2^{\bar{1}} \cdot 1_3 \vee a_0^{\bar{1}} \cdot a_1^{\bar{1}} \cdot b_2^{0} \cdot 0_7 \vee a_0^{\bar{1}} \cdot a_1^{\bar{1}} \cdot b_2^{1} \cdot \bar{1}_6$$
$$\vee a_0^{0} \cdot 0_5 \vee a_0^{1} \cdot b_4.$$

图 12(a) 便是所设计的一位乘法器阵列，实际上它只占用 3 个细胞 5 个边沿输入。

同理图 12(b) 和 (c) 分别是一位全加器和及其进位输出的阵列，它们一共占用了 10 个细胞，21 个边沿输入，其中 $\vec{C}^* = C^{*\bar{1}} \cdot 0 \vee C^{*0} \cdot 1 \vee C^{*1} \cdot \bar{1}$，$\vec{C}^* = C^{*\bar{1}} \cdot 1 \vee C^{*0} \cdot \bar{1} \vee C^{*1} \cdot 0$ 两个运算是由单独的两个细胞完成的。

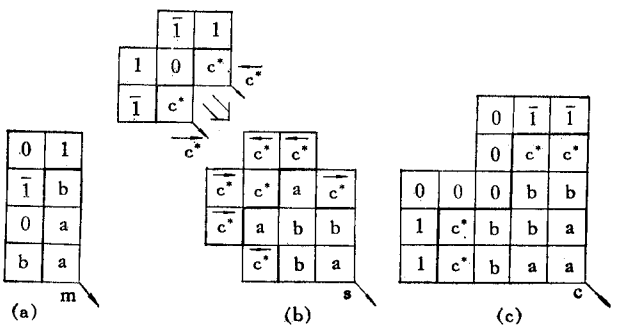


图 12　一位三进制乘法器和全加器阵列

(二) 四位高速三进制乘法器阵列

关于各类型运算器理论曾发表过许多文章，大多是围绕着提高运算速度、多功能、减少电路和有利于大规模集成化等方面为目的的。1977 年 Hamacher 曾经作过这样的设计，它采用与二进制乘法器相类似的算法，设计了各种不同类型的三进制乘法器。结果证明：16 位对称编码的三进制乘法器比 24 位二进制乘法器输入端数减少 80%[14]。

本文又把 Agrawal 最近发表的四位二进制高速乘法器阵列[2]修改为按本文定义的四位三进制高速乘法器阵列(见图 13)。阵列中每一个运算器细胞具有相同的功能（见图

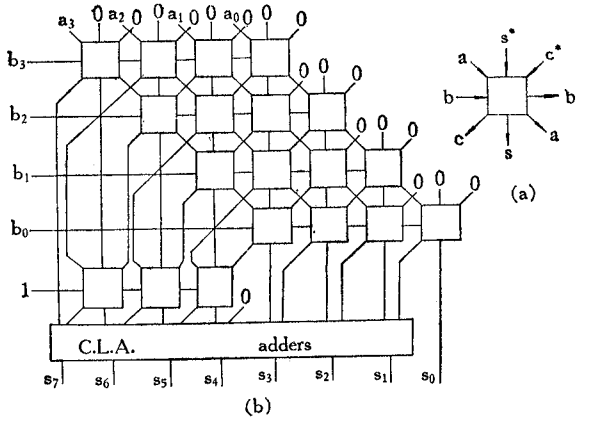


图 13 (a) 基本的运算器细胞 (b) 四位三进制乘法器阵列

13(a)),并有如下输入输出关系:

$$\left.\begin{aligned} S &= (S^* + C^* + a \times b)_{\text{mod}-3} \\ C &= (S^* + C^* + a \times b)_{\text{carry}-3} \end{aligned}\right\} \quad (6)$$

其中 S^*、C^*、a 和 b 分别是前位和、进位输入、被乘数和乘数. 实际上它们就是一位带乘法器的三进制全加器阵列. 所以每一个运算器细胞可用如图 12 所设计的阵列实现它.

图 13(b) 则是用上述运算器细胞按半规律方式排列构成的四位三进制高速乘法器阵列. 它与 Agrawal 具有相同的进位存储连接方式. 通过快速的先行进位加法器 ($C.\ L.\ A$) 加上最高两位和数,从而降低了乘法器的时延.

五、比 较

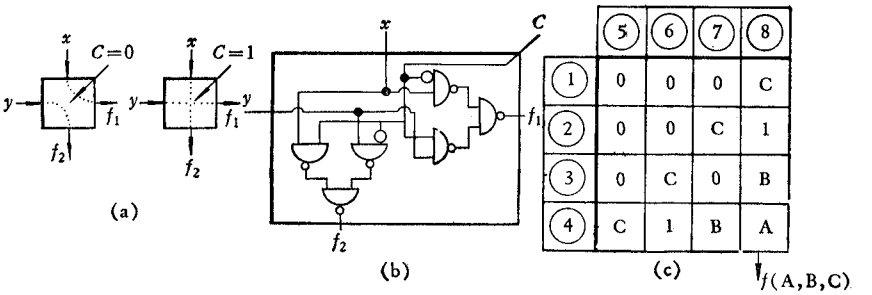
(一) 与 Kukreja 的二值胞腔阵列比较[1]

图 14(a) 是 Kukreja 定义的二值细胞,每一个细胞的布尔表达式为:

$$f_1 = \bar{C} \cdot x \vee C \cdot y, \quad f_2 = C \cdot x \vee \bar{C} \cdot y \quad (7)$$

图 14(b) 是每一个二值细胞实现的框图. 显然它必须使用 6 个双头与非门和 2 个反相器. 图 14(c) 则是 Kukreja 设计的三变元二值逻辑通用阵列. n 变元的二值逻辑通用阵列必须使用二值细胞的总数目为:

$$\theta = (2^{2n-2}) \quad (8)$$



	5	6	7	8
1	0	0	0	C
2	0	0	C	1
3	0	C	0	B
4	C	1	B	A

图 14 Kukreja 的二值细胞

根据二值细胞的特征式(7)和三值细胞的特征式(1),又根据 Mouftah[10] 的三值 CMOS 开关电路或本文图 3 的 I^2L 三值细胞电路,一个 Kukreja 的二值细胞与一个三值细胞使用电路数目之比值约为 3:2. 以实现相同组合数输出来比较,根据式(5)和式(8)的计算,得到如表 5 所示的比较结果. 从表中看出: 采用三值逻辑胞腔阵列比二值逻辑胞腔阵列使用电路减少 80%,功耗降低 60%,速度提高 50%,总信息流量增加 30%,而且随着 n 变

元的增加，这个比值的变化是令人鼓舞的.

表 5. 通用阵列的比较

需要量 / 通用阵列	细胞		电路		时延		功耗
	总数	比例	总数	比例	门级	比例	比例
2 变元三值逻辑 3 变元二值逻辑	10 16	1 1.6	80 192	1 2.4	12 14	1 1.1	1 1.6
3 变元三值逻辑 5 变元二值逻辑	91 256	1 2.8	728 3072	1 4.2	39 62	1 1.6	1 2.8

(二) 与 Agrawal[2] 的高速乘法器阵列比较

文献[2]中的图 1 定义每一个二进制运算器细胞具有如下特征式：

$$S = a \oplus c_1 \oplus [bp], \quad C_2 = ac_1 + [a + c_1]bp \tag{9}$$

其中 a、c_1、b 和 p 分别为前位和、进位输入、被乘数和乘数. $\oplus$为模二加，$+$为逻加而 bp 和 ac_1 则为二元逻乘.

如果每一个二进制运算器细胞采用 Kukreja 的阵列实现它，根据式(9)和图14(c)，至少要用32个细胞. 另一方面，如果本文设计的三进制运算器细胞采用如图 12 所示的阵列实现它，只需要 23 个细胞. 根据一个 Kukreja 的二值细胞需要电路数目为三值细胞的 1.5 倍的估计，那么一个二进制运算器细胞的电路数目便是三进制运算器细胞的 2 倍，所以一个四位二进制乘法器的设备量便是一个四位三进制乘法器的设备量的 2 倍，显然，即使它们的设备量相同，这个结果也有力地证明了：无论采用规律或半规律的排列方式，对称开关逻辑的三值胞腔阵列将可能弥补二值胞腔阵列必须使用太多电路的缺陷，从而使胞腔阵列的研究更有实现的可能.

参　考　文　献

[1] S. N. Kukreja and I. N. Chen, *IEEE Trans. on Computer*, C-**22** (1973), 813.
[2] D. P. Agrawal, *IEEE Trans. on Computer*, C-**28** (1979), 215.
[3] S. B. Akers, *IEEE Trans. on Computer*, C-**21** (1972), 848.
[4] D. C. Rine, Computer Science and Multiple-valued Logic Theory and Applications, *Chapt.* 14, (1977), 397—419.
[5] Z. G. Vranesic, *IEEE Trans. on Computer*, C-**26** (1977), 1181.
[6] 樋口龙雄，数理科学， 1980 年，第 200 期，第 55 页.
[7] 金晓龙，楼世博，铁道学报 1979 年，第 1 期，第 55 页.
[8] 郑启伦，电子计算机动态， 1978 年，第 12 期，第 26 页.
[9] 郑启伦，电子学报， 1980年，第 2 期，第 62 页.
[10] H. T. Mouftah and I. B. Jordan, in Proc. Int. Symp. on Multiple-Valued Logic, (1974), 285—302.
[11] H. T. Mouftah and I.B. Jordan, *IEEE Trans. on Computer*, C-**26** (1977), 281.
[12] D. Etiemble and M. Israel, *IEEE Trans. on Computer*, C-**26** (1977), 1222.
[13] T. T. Dao, E. J. McCluskey and L. K. Russell, *IEEE Trans. on Computer*, C-**26** (1977), 1233.
[14] D. C. Rine, Computer Science and Multiple-valued Logic Theory and Applications, *Chapt.* 17 (1977).