第6卷 第4期 自 动 化 学 报 Vol. 6, No. 4
1980 年 10 月 ACTA AUTOMATICA SINICA Oct., 1980

626 型快速傅里叶变换处理机方案介绍*

康继昌 徐乃平 洪远麟 吕传生
(西 北 工 业 大 学)

摘 要

快速傅里叶变换（FFT）处理机是最近十年内发展起来的。我们曾为天津电子仪器厂和国家海洋局五所的 FA1 型实时信号分析仪进行了方案设计，已于 1979 年研制成功。该机适宜于处理音频信号，1024 点的 FFT 处理时间为 40 ms。为了满足更高频信号的要求，我们正和上海 573 厂共同研制速度更快的 626 型处理机，1024 点的 FFT 处理时间约为 5 ms，现正进行总调。由于国内已有不少单位介绍过这类处理机的设计[1]，因此本文仅就 626 机如何提高处理速度的问题作重点介绍。

一、主 要 性 能

1. 功能： FFT 频谱、反变换、自功谱、互功谱、自相关、互相关等.

2. 处理速度： 1024 点的 FFT 处理时间约为 5 ms，即可实时分析的信号带宽约为 100 kHz（采样频率为 200 kHz）.

3. 输入通道： 2 路，动态范围为 60 分贝(二进制编码 10 位).

4. 输出： 配示波器和 X-Y 记录仪，也可以和 DJS-130 机、622 机等计算机联机工作.

5. 机器字长： 16 位.

二、基 4 算 法

成倍地提高元器件的速度是有一定困难的. 采用基 4 算法也是一种提高处理速度的方法.

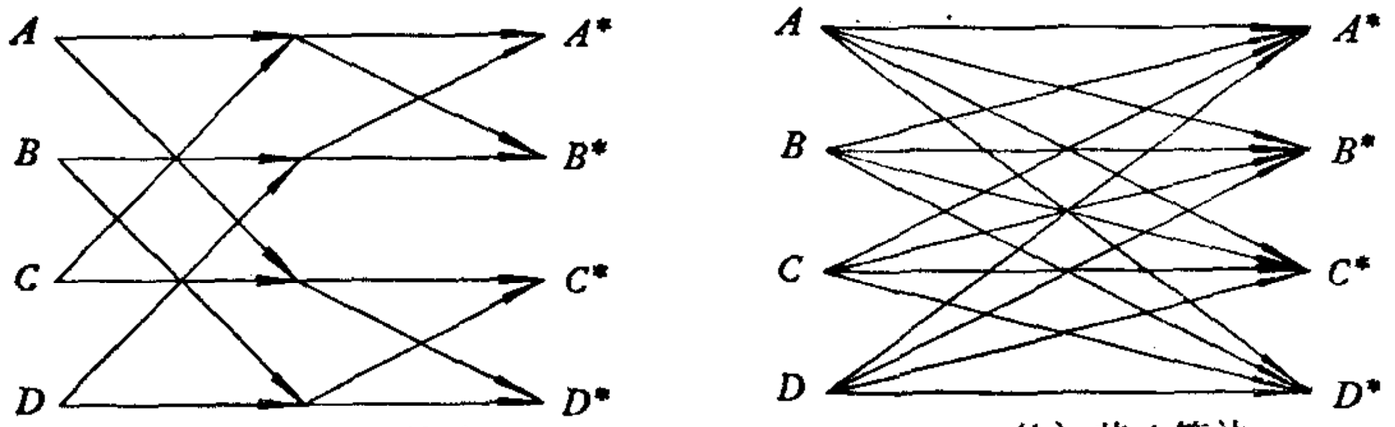
见图 1，1 个基 4 的蝶形运算相当于 4 个基 2 蝶形运算.

我们以 $N = 16$ 为例，来说明基 4 算法[2].

令 $e^{-j2\pi/16} = W$，则离散傅里叶变换可写为:

---

* 本文曾在 1979 年全国计算机应用学术交流会和昆明全国计算机年会上宣读. 本文修改稿于 1980 年 4 月 7 日收到.



(a) 基 2 算法　　(b) 基 4 算法

图 1　FFT 算法信号流程图

$$X(n) = \sum_{k=0}^{15} x(k)\, W^{nk} \tag{1}$$

其中 $n, k = 0, 1, 2, \cdots, 15$

为了实现基 4 算法，变量 n, k 用 4 进制数来表示：

$$n = 4n_1 + n_0, \quad n_1, n_0 = 0, 1, 2, 3$$

$$k = 4k_1 + k_0, \quad k_1, k_0 = 0, 1, 2, 3$$

于是(1)式变成：

$$X(n_1, n_0) = \sum_{k_0=0}^{3} \left[\sum_{k_1=0}^{3} x(k_1, k_0)\, W^{4nk_1} \right] W^{nk_0} \tag{2}$$

利用旋转因子 W 的周期性，即 $W^{16} = 1$，可以化简：

$$W^{4nk_1} = W^{4(4n_1+n_0)k_1} = [W^{16}]^{n_1k_1}\, W^{4n_0k_1} = W^{4n_0k}$$

代入(2)式，即得基 4 算法中的内层和式：

$$x_1(n_0, k_0) = \sum_{k_1=0}^{3} x(k_1, k_0)\, W^{4n_0k_1} \tag{3}$$

外层和式为：

$$X(n_1, n_0) = \sum_{k_0=0}^{3} [x_1(n_0, k_0)\, W^{n_0k_0}]\, W^{4n_1k_0} \tag{4}$$

为直观起见，(3)式可写成矩阵形式：

$$\begin{bmatrix} x_1(0, k_0) \\ x_1(1, k_0) \\ x_1(2, k_0) \\ x_1(3, k_0) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^4 & W^8 & W^{12} \\ W^0 & W^8 & W^{16} & W^{24} \\ W^0 & W^{12} & W^{24} & W^{36} \end{bmatrix} \begin{bmatrix} x(0, k_0) \\ x(1, k_0) \\ x(2, k_0) \\ x(3, k_0) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} x(0, k_0) \\ x(1, k_0) \\ x(2, k_0) \\ x(3, k_0) \end{bmatrix} \tag{5}$$

同样，(4)式也可写成矩阵形式：

$$\begin{bmatrix} X(0,n_0) \\ X(1,n_0) \\ X(2,n_0) \\ X(3,n_0) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} x_1(n_0,0) \\ x_1(n_0,1)\,W^{n_0} \\ x_1(n_0,2)\,W^{2n_0} \\ x_1(n_0,3)\,W^{3n_0} \end{bmatrix} \tag{6}$$

(6)式比(5)式具有更普遍的形式,每次取 4 个操作数,经过运算得 4 个计算结果. 这个过程称为一个基 4 的蝶形运算. 为方便起见, 4 个操作数分别表示为 A, B, C, D; 4 个计算结果分别表示为 A^*, B^*, C^*, D^*; 把与 B, C, D 相对应的旋转因子 W^{n_0}, W^{2n_0}, W^{3n_0} 分别表示为 W^B, W^C, W^D. 于是(6)式可简化为:

$$\begin{bmatrix} A^* \\ B^* \\ C^* \\ D^* \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} A \\ BW^B \\ CW^C \\ DW^D \end{bmatrix} \tag{7}$$

或展开如下:

$$\left.\begin{aligned} A^* &= (A + CW^C) + (BW^B + DW^D) \\ B^* &= (A - CW^C) - j(BW^B - DW^D) \\ C^* &= (A + CW^C) - (BW^B + DW^D) \\ D^* &= (A - CW^C) + j(BW^B - DW^D) \end{aligned}\right\} \tag{8}$$

可见,每个基 4 蝶形运算只有 3 次复数乘法,比用基 2 算法减少 1 次复数乘法,因此总的乘法次数就可以减少 25%.

N 点基 4 算法的迭代次数为 $\log_4 N$, 而基 2 算法的迭代次数为 $\log_2 N$. 我们知道 $\log_4 N = \frac{1}{2}\log_2 N$, 即基 4 算法的迭代次数可以减少一半. 每迭代一次要访问一次内存的数据. 迭代次数减少一半,就意味着访问内存的次数减少一半. 在其它条件相同的情况下,速度就可提高一倍.

当然,基 4 算法对变址和控制部件会带来一定的复杂性. 我们采取了一定的措施加以解决.

三、MOS 随机存贮器

本机随机存贮器有 3 个存贮体: 1个是输入缓冲, 1 个是输出缓冲, 1 个是运算的中间结果. 每个体容量为 1K 字. 输入体字长为 10 位,其它两个体字长为 16 位. 3 个体要求能够并行工作. 有时还要求 32 位字长工作,即实部、虚部各 16 位同时存取. 如采用磁芯存贮器很难满足这些要求,速度也差的很远.

我们采用上海无线电 14 厂的 CP 1603 MOS 动态存贮器. 该存贮器读、写周期都小于 600 ns, 我们工作在 750 ns,以便留有余地.除了速度快以外,它的主要优点是不破坏读出,选址灵活,因而能满足上述的各种特殊要求. 缺点是集成度不太高,每片只能存贮256个代码,价格较贵. 但本机随机存贮器容量不是太大, 共用 MOS 组件 168 片, 价格在 2 万元左右,显然是可行的.

四、运　算　器

在运算器中，主要是做复数乘法，例如：

$$(a + jb)(c + jd) = (ac - bd) + j(ad + bc) \tag{9}$$

可见，实部和虚部都是两个乘积的代数和．针对这个特点，我们设计了两套双加法器，分别计算实部和虚部．

被乘数可用 $M + jM'$ 表示，乘数是旋转因子，它有如下形式：

$$\cos\frac{n}{N}2\pi + j\sin\frac{n}{N}2\pi$$

以实部为例，计算式为：$M\cos\frac{n}{N}2\pi - M'\sin\frac{n}{N}2\pi$

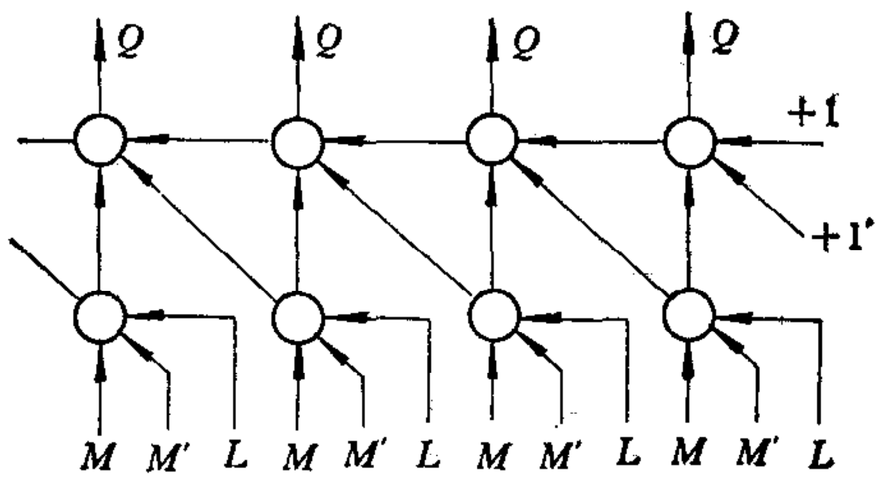


图 2　双加法器

我们采用补码两位乘法规则，三角函数值取 10 位，作为乘数，故 5 拍可以完成一次乘法．

如果只做一个乘法，用一个加法器就够了．例如做 $M\cos\frac{n}{N}2\pi$ 时，可用 cos 函数值控制 M 的累加．现在一次要完成实部的 2 个乘法，可用一个双加法器，如图 2，图中主要是两排全加法器，L 是部分积（最后是乘积），M，M' 是被乘数的实部和虚部，分别为乘数 cos，sin 函数所控制．第二排全加器只需 2 级门延迟时间，第一排全加器需分组快速进位链，最多延迟时间为 6 级门．所以双加法器共有 8 级门延迟时间．采用延迟时间为 10 ns 的高速 TTL 电路时，总延迟时间为 80 ns．

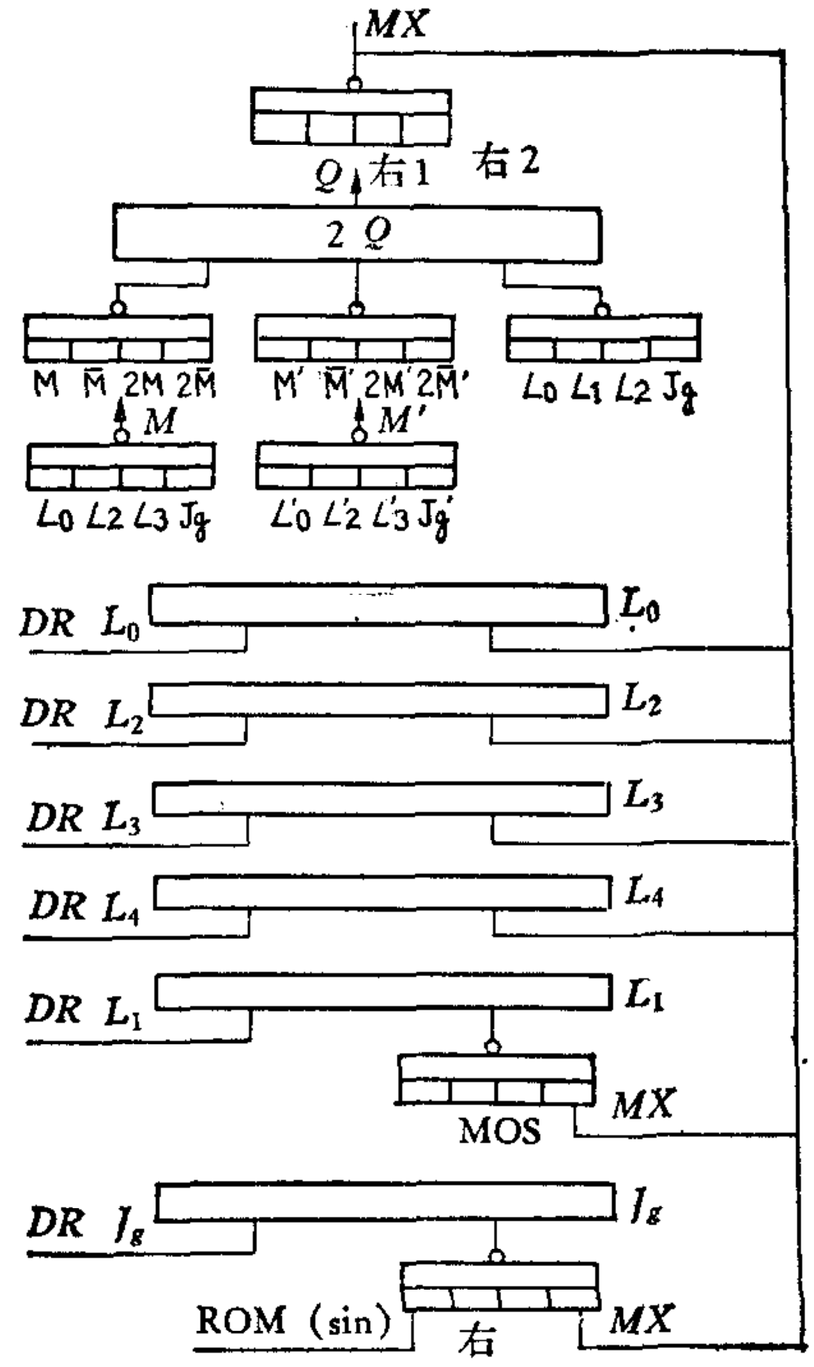


图 3　运算器框图

图 3 为实数部分的运算器框图，虚数部分与此类似．

图 3 中 L_0—L_4 为数据寄存器，都可存放中间计算结果，其中 L_1 还可接受从 MOS 存贮器来的数据．为了防止溢出，故有多路开关控制右移 1 位或 2 位．从正弦函数固定存贮器来的函数值送入 J_g 寄存器作为乘数，做乘法时控制被乘数的累加．M 为被乘数多路开关，以便各有关数据寄存器均可作为被乘数．哪一个数据寄存器作为部分积也有多

路开关进行控制. 母线多路开关 MX 可控制部分积右移 1 位或 2 位.

本来计算一个以 4 为基的蝶形运算需时 12 μs，但适当安排时间，可使前后两个蝶形运算的一部分时间重迭起来，这样平均每个蝶形运算的时间只要 6 μs.

一个蝶形运算的前一半流程图如图 4. 取数或存数时间为 1 大拍，即 750 ns. 每 1 大拍又分为 3 小拍，每小拍为 250 ns，可以完成一次加法，即每秒可以做 400 万次加法. 由于实部、虚部共有 4 套加法器在并行运算，故总的计算速度相当于每秒 1600 万次加法.

蝶形运算的计算公式如下：

$$b_1 = BW^B + DW^D \qquad b_2 = b_1 - 2DW^D$$

$$a_1 = A + CW^C \qquad a_2 = A - CW^C$$

$$A^* = a_1 + b_1 \qquad C^* = A^* - 2b_1$$

$$D^* = a_2 + jb_2 \qquad B^* = D^* - 2jb_2$$

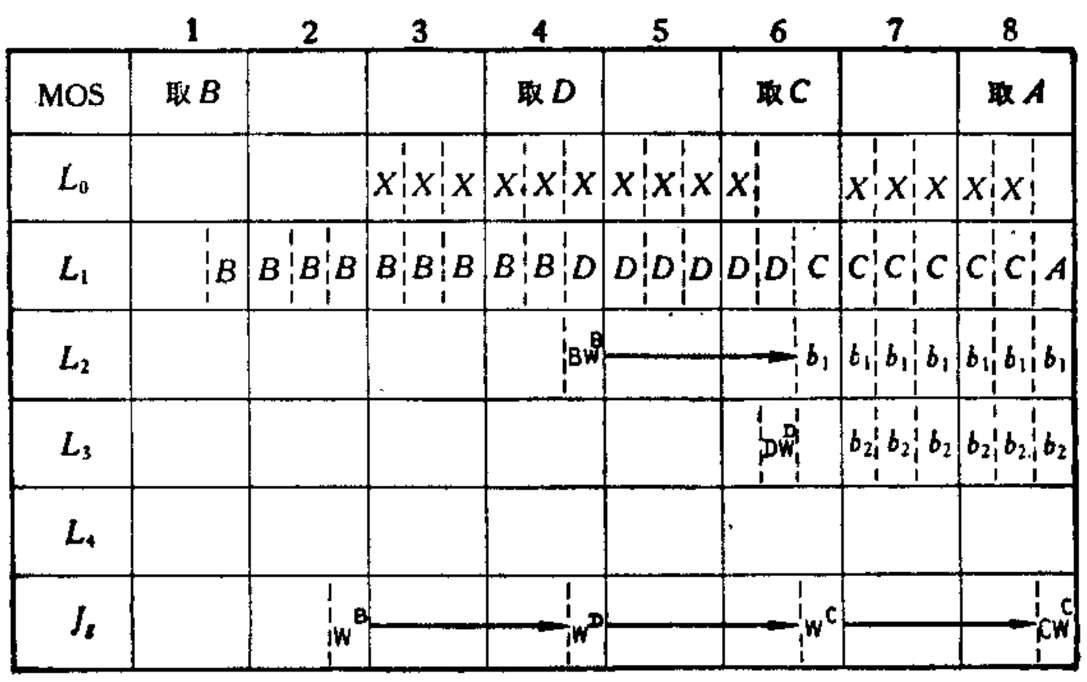


图 4 前一半流程图

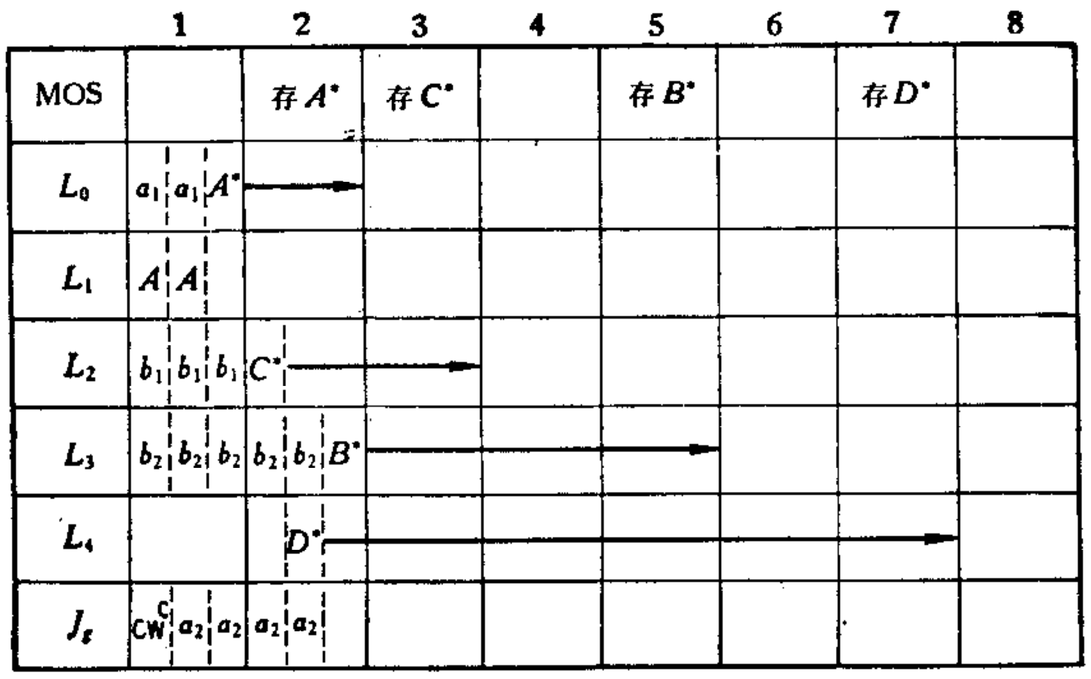


图 5 后一半流程图

现在解释一下图 4：第 1 大拍取 B，到第 3 小拍时 B 到 L_1. 第 2 大拍取三角函数到 J_g 中. 第 3 大拍开始做乘法 BW^B，部分积放在 L_0，5 小拍做完，在第 4 大拍的末尾乘积放到 L_2. 第 4 大拍还取 D、W^D，接着做乘法. 同样经过 5 小拍，乘积 DW^D 放在 L_3. 第 6 大拍的第 2 小拍计算 b_1，结果在 L_2；接着计算 b_2，结果在 L_3；同时取 C 和 W^C. 同样再经过 5 小拍得乘积 CW^C，放在 J_g. 第 8 大拍还取 A，放在 L_1.

一个蝶形运算的后一半流程图见图 5. 第 1 大拍的第 1 小拍得计算结果 a_1 在 L_0，第 2 小拍得 a_2 在 J_g，第 3 小拍得 A^* 在 L_0. 第 2 大拍把 L_0 中的计算结果 A^* 存

入 MOS 存贮器，同时还计算 C^*，D^*，B^* 依次放在 L_2，L_4，L_3 中．第 3 大拍存 C^*，第 5 大拍存 B^*，第 7 大拍存 D^*，一个蝶形运算就此结束．下一个蝶形运算又重复进行，但对照图 4 和图 5 可见，各个部件在时间上恰好错开，所以下一个蝶形运算可在前一个蝶形运算的后半流程中同时进行．就是说，在图 5 的第 1 大拍中可以取 B，第 2 大拍中可以取 W^B，第 3 大拍中可以做乘法，依此类推．这就实现了重迭计算，如下所示：

第 1 个蝶形

第 2 个蝶形

第 3 个蝶形

五、变 址 部 件

由基 4 算法可知，参加蝶形运算的数据要按计算流程的次序进行存取．通常并非完全按顺序进行存取，而是按一定的规律变化，故须设计专门的变址部件进行控制．为此，必须了解变址的规律．

每个蝶形有 A，B，C，D 4 个数据，分别有 4 个地址，依次称为 A 地址，B 地址，C 地址，D 地址．变化规律如下：

第　一　遍

A 地址	B 地址	C 地址	D 地址
0 0 0	1 0 0	2 0 0	3 0 0
0 0 1	1 0 1	2 0 1	3 0 1
0 0 2	1 0 2	2 0 2	3 0 2
0 0 3	1 0 3	2 0 3	3 0 3
0 1 0	1 1 0	2 1 0	3 1 0
. . .	. . .	. . .	. . .

第　二　遍

A 地址	B 地址	C 地址	D 地址	
0 0 0	0 1 0	0 2 0	0 3 0	
0 0 1	0 1 1	0 2 1	0 3 1	
0 0 2	0 1 2	0 2 2	0 3 2	
0 0 3	0 1 3	0 2 3	0 3 3	
↙	↙	↙	↙	跳跃进位
1 0 0	1 1 0	1 2 0	1 3 0	
. . .	. . .	. . .	. . .	

地址用 4 进制数表示，计数时逢 4 进位．

第一遍：第 1 位为“控制位”，0 表示 A 地址，1 表示 B 地址，2 表示 C 地址，3 表示 D 地址．除第 1 位以外，其余各位按 4 进制顺序计数．

第二遍：第 2 位为“控制位”，同样以 0，1，2，3 分别表示 A，B，C，D 地址．除第 2 位(即控制位)外，其余各位也按 4 进制顺序计数．但是进位要跳过控制位，故称跳跃进位．其余各遍可依次类推．

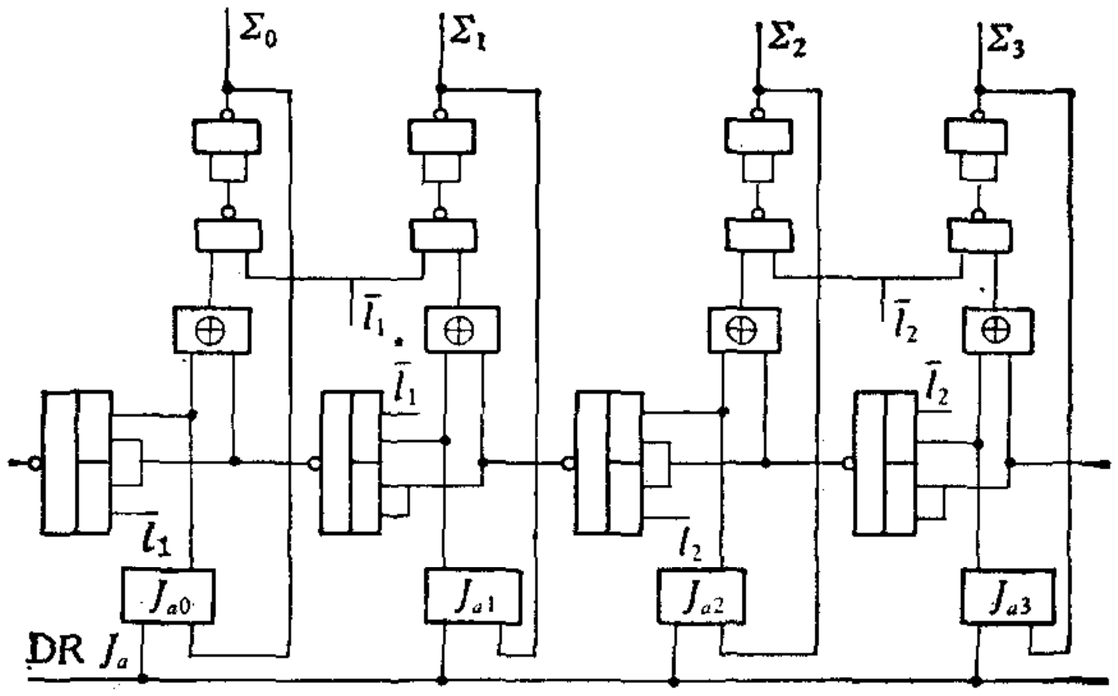


图 6　跳跃进位计数器

图 6 是跳跃进位计数器，它由交替进位线路附加一些跳跃进位的控制线路所组成．J_a 为基址寄存器．$l_1, l_2, \cdots$是遍数控制信号，第 1 遍时 l_1 为高电位，第 2 遍时 l_2 为高电位，……．第 1 遍时，l_1 开启跳跃进位门，封锁 Σ_0，Σ_1 的输出．第 2 遍时，l_2 开启跳跃进位门，封锁 Σ_2，Σ_3 的输出，依次类推．每发一次 DRJ_a 脉冲，便把下一个基址打入 J 寄存器．

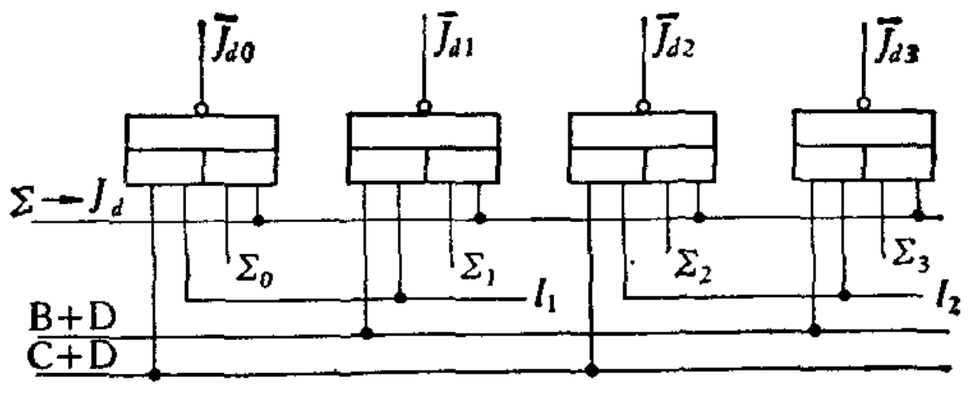


图 7　数据地址形成线路

图 7 为数据地址形成线路．$\Sigma \rightarrow J_d$ 控制信号传送 A 地址，传送 B 地址时，还需加 $(B+D)$ 信号，使单数号地址送 1．传送 C 地址时，则加 $(C+D)$ 信号，使双数号地址送 1．同时加 $(B+D)$ 和 $(C+D)$ 信号，则送 D 地址．当然，这些信号还要受 l_1，l_2，…的控制．例如，在第 2 遍时，Σ_2，Σ_3 两位为 0，即表示 A 地址．加 $(B+D)$ 信号时控制位为 1，送 B 地址．加 $(C+D)$ 信号时控制位为 2，送 C 地址．两者都加时控制位为 3，送 D 地址．其余各位直接传送 Σ 值．

六、控 制 器

本机采用微程序控制，微程序共约 500 条微指令，每条微指令字长约 60 位，均存放于变压器穿线式固定存贮器中．每 1 小拍 250 ns，要有 1 条微指令进行控制．由于一个变

压器穿线式固定存贮器难以达到这样高的速度，我们采用两个 500 ns 读数周期的固存交替工作来解决这个困难. 当然,以后有条件时可考虑采用半导体固定存贮器.

从计算机系统结构的观点来看,本机与通常的存贮程序式计算机不同,因为它没有程序，也不用程序进行控制. 它可以说是一台微指令流进行控制的机器. 从三个存贮体中的数据流来看,数据从输入体到迭代体(中间计算结果),再到输出体,是流水式的. 但是,从运算器的实部、虚部同时进行运算来看,又是并行方式的. 总之,从系统结构来看,本机是根据 FFT 算法的特点,灵活设计的.

参　考　文　献

[1] 康继昌、徐乃平，FFT 硬件处理机,西北工业大学学报 (1978 年) 第 3 期,第 17—24 页.
[2] E. O. 布赖姆,快速傅里叶变换,上海科学技术出版社(1979)第 210 页.

AN INTRODUCTION TO THE PROJECT OF 626-TYPE FFT PROCESSORS

KANG JI-CHANG　XU NAI-PING
HONG YUAN-LIN　LÜ CHUAN-SHENG
(Xibei Industrial University)

ABSTRACT

FFT processors have been developed within the recent ten years. The National Ocean Bureau and Tianjin Electronic Instruments Factory succeeded in developing a real-time signal analyzer FA-1 type in 1979, based on our schematic design. That machine is used for processing acoustic signals. Its computing time of 1024-point FFT frequency spectrum is about 40 ms. Now, we are developing a 626-type FFT processor to satisfy the requirement of higher frequency signals. Its computing time of 1024-point FFT frequency spectrum is about 5 ms. As there have already been papers introducing projects of FFT processors, our paper will only show essentially how the high speed is achieved.