对话式电路设计系统中电路图构成及光笔检索的一种方法

罗 伯 鹏

提 要 本文讨论利用数据结构中所存贮的连接点的行、列定位信息，使电路图重显或平移、放大以及对电路图进行光笔检索和修改的一种算法。

一、引 言

显示于屏面上的电路图，只是设计人员将电路描绘给计算机的一种手段。图形的来源可以是纸带输入（例如采用 GCAP2 的编码法[1]），也可以是直接用光笔、键盘逐步构成[2]；或两者兼而有之，例如在纸带输入后用光笔进行修改。计算机能“知道”送入的是什么电路，并响应任意的修改，最后仍能形成正确的分析计算方程的关键是利用了在构图时所形成的数据结构。

在显示屏上显示电路常采用等距网格法；即将显示屏划分为若干正方网格，可在某些网格点上显出电路图的联接点(元件与元件或与导线之间的联接点以下简称为接点)，在两接点之间联以垂直或水平元件[3]。采用这种办法时，对器件显然不能直接使用其等效电路而只能使用记号，否则在屏上就只能显示很少几个器件的等效电路图。这样，便出现了将器件的等效电路图描绘给计算机并赋以适当记号的问题。此外，在同一显示屏上还需要显示分析计算的结果如解曲线等。因此，显示画面不可避免地要经常更换：有时显示全局电路图，有时是器件电路图，有时则是解曲线。特别当使用的器件种类繁多或还要不断地依据解曲线修改电路时，这种更换就会更加频繁。

显示的电路图需要经常变动，还由于显示屏小；于是为了将一个大型电路逐步用光笔输入或为了观察一个大型电路，就需要将屏上显出的电路图不断向上下或左右平移；为了使标注于元件旁的数值（如直流分析计算结果）能清晰显示，便需要将电路图予以开窗放大。对于电路图的这些必要的更换或变动，不可能也不必要在内、外存中保存各种电路图的显示档案。事实上，如果在保存接点拓扑结构信息的同时，还保存它的定位行号(I)及列号(J)，则关于电路图的更换与变动是容易实现的。而且，利用(I，J)信息，还容易解决光笔击中子图形的问题。

综上所述可以看出：保存电路的物理信息、拓扑信息以及接点定位信息的数据结构在对话式电路设计系统中的重要作用。这还可以用图解表述如下：（图 1 中，“电路图的建立”系对纸带输入而言）。

本文1982年5月4日收到

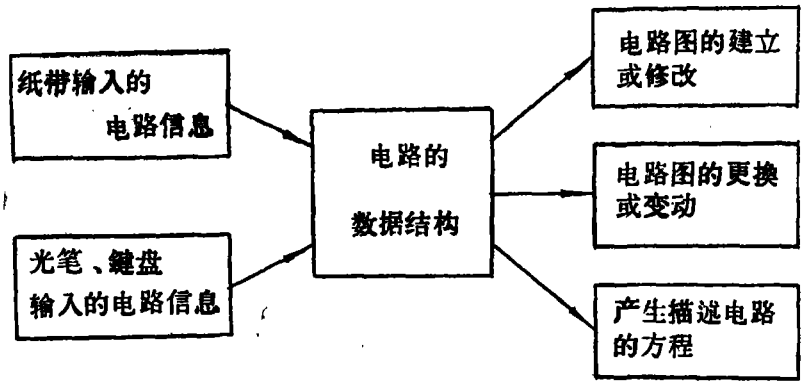


图 1

关于由数据结构产生方程的问题拟另行讨论，本文仅讨论与图形有关的问题，但在具体应用时则还需根据情况增加和修改一些细节。

二、所采用的数据结构简介

由于所显示的元件只能是垂直或水平的，故每个接点最多只能联 4 个元件。如所联元件超过 4 个或要在一个接点沿同一方向并联两个以上的元件，则应使用两个以上的接点其间联以导线。因此，本文所称的接点，可能与电路理论中的节点相同也可能不同；在接点之间联有导线时，则所有用导线联起来的接点才相当于一个节点。

所采用的记录类型包括："接点"、"元件"、"导线"、"器件记号"（或"器件使用"型记录）。而在器件等效电路中，还需增加"器件头"（或"器件等效电路引导块"）。在全局电路数据结构中，通过"器件记号"记录可索引到"器件头"记录；这样，在构成方程时，就可将该器件的等效电路嵌入到全局电路中来处理。接点与元件记录示如图 2。在表征全局电路或器件等效电路的数据结构中，接点（包括器件记号）记

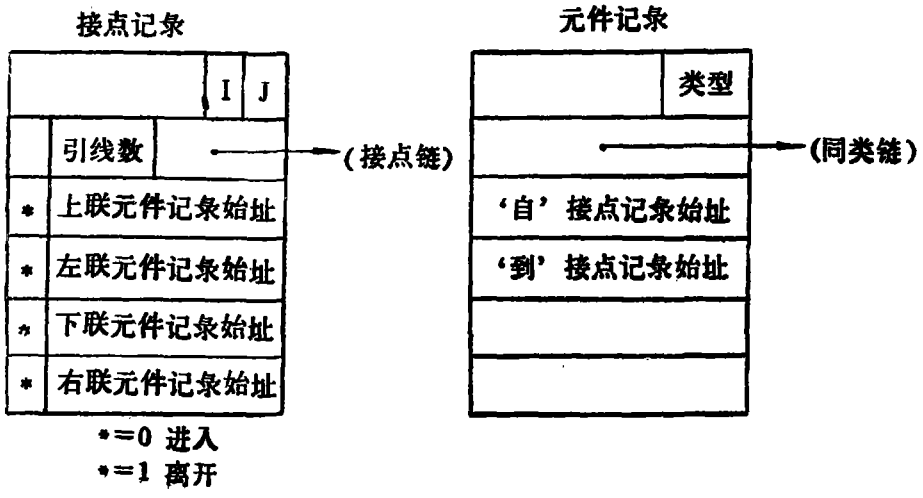


图 2
(图中空域用以填写其他信息)

录借助于单向指针形成接点链；而接点与所联接的元件形成小环。数据结构的大致情况如图3所示。此图表示元件 El 自接点 N_1 联到接点 N_3，接点 N_1 的下联元件、N_3 的上联元件则均为 El。

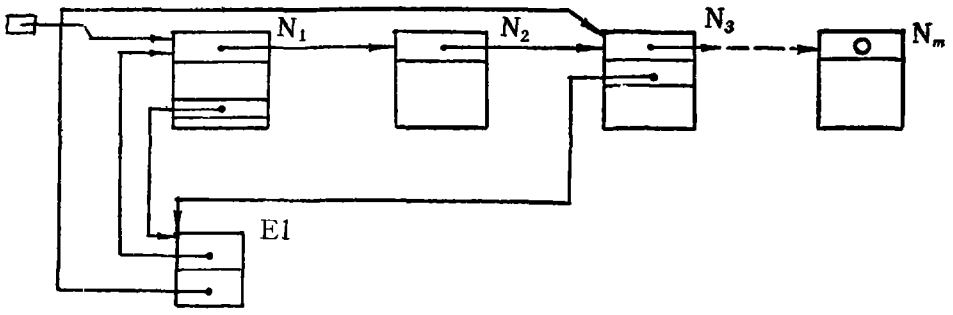


图 3

左上角网格点的屏坐标为 (x_0, y_0)，其行、列号则为 I_0，J_0。机器预置 $I_0=0$，$J_0=0$，亦可由用户规定初值。每当全图向左平移 p 个间距、向上平移 q 个间距时，则执行：

$$I_0+p\Rightarrow<I_0>,\quad J_0+q\Rightarrow<J_0>$$

此处 p 或 q 若为负值则表示反向平移。显示一行(或一列)的网格点数 n 也系由机器预置或由用户规定。故右下角网格点的行、列号为 I_0+n，J_0+n；设其屏坐标为 (x_n, y_n)，则接点间距 d 由下式给出：

$$d=\left[\frac{x_n-x_0}{n}\right]$$

I_0、J_0、d 是重要的作图参数。元件子图形的尺寸应随 d 取值的变化而有所调整。

构图时将在 $x=x_0+(I-I_0)d$，$y=y_0+(J-J_0)d$ 处显示一个指示接点定位的软件游标。软件游标的位置可由4个按键（各表示向左或下或右或上移动一个间距）随意调动；故当送入接点定位請求时，便极易取得该接点的 (I, J) 值。但若用纸带输入时，则 GCAP2 法中的节点编码需改变为接点的行号与列号。

接点链是按 I 值升序（同一行中则按 J 值升序）整理的。即在构图时新链入的接点记录应按 I、J 的升序插入接点链中。

三、从数据结构显示电路图的算法

为构成电路图的显示档案，应首先追踪接点链：对于每个接点或器件记号，取出其 (I^*, J^*) 信息算出定位坐标 (x^*, y^*)，即可构成定位点，而后调用接点或器件记号的子图形即可；对于接点所联元件，则需判明元件类型，而后按不同类型调用不同的元件子图形。由于接点记录是按 (I, J) 升序排列的，故对每个接点仅需取出其右联和下联元件来显示。为避免产生重复图形，不应考虑上联和左联元件；特别当接点为右上角点时只取下联元件，为左下角点时只取右联元件，为右下角点则全部不取。

元件子图形的画法应作如下规定：对于水平元件恒从左端画到右端，对于垂直元件则恒从上端画到下端。这样，元件子图形定位点的取法便是确定的：对于右联元件，定位点取为 (x^*+d', y^*)，对于下联元件，则定位点取为 (x^*, y^*+d')。d' 的意义可参看图 6 。

根据以上讨论，可将追踪接点链构成电路图的算法用框图表出如下：

(1) 主程序：见图 4 。

(2) 接点与元件显示子程序

下述框图（图 5 ）中，「显 (I, J) 接点」，「显右联元件」，「显下联元件」均是构成有关显示档案的子程序。显然，若接点的右（或下）联元件指针域的内容为 0 ，则该有关子程

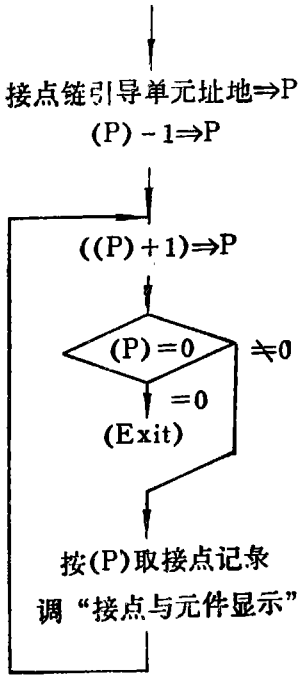


图 4

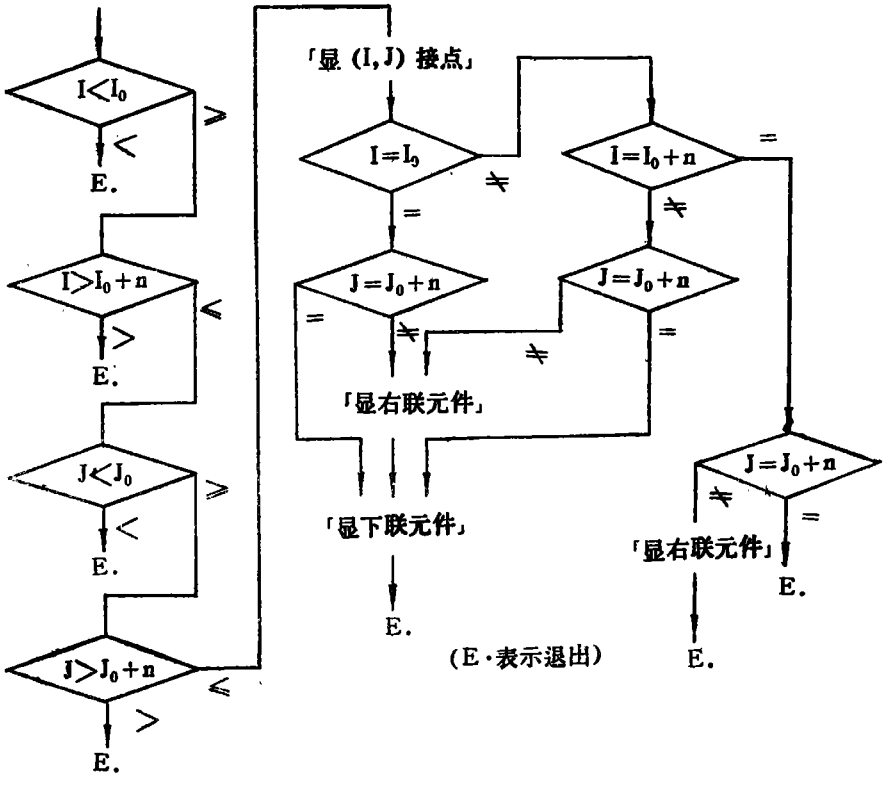


图 5

序返回。其中用到的 I，J 信息则是从接点记录的相应域中取出来的。对于大型电路，可采用行始接点地址索引表，这样能加快处理速度。

(3) 平移与放大

在调用上述主、子程序之前，若先改变 I_0，J_0 之值则将发生平移；若事先改变 n 或改变 d 之值则将发生放大或缩小。显然程序本身不需要进行修改。

四、光笔指点于显示对象后检索相应数据记录的算法

光笔指点于显示对象时，由光笔中断送入计算机的主要信息通常为：被击中对象的显示缓存地址 B^* 及其坐标 x^*, y^*。（若击中线段，则 x^*, y^* 为此线段终点坐标）。通常根据 B^* 查对象的缓存始址表便能确定所击中的对象是哪一个。但现在的问题在于：在电路图中，显示的元件或器件记号都是调用的子图形。例如图中有 17 个电阻元件图，无论光笔击中哪一个所得 B^* 值都是属于同一子图形因而是一样的，故从 B^* 值程序就无从辨别所击中的是屏上的哪个电阻。下面介绍的“I-J 算法”，其基本思想是由屏坐标 x^*，y^* 求出 I，J 值后，由之找相应接点再找元件。当击中对象为接点时，求出的 I，J 值即是该接点的 I，J。当击中对象为水平（或垂直）元件时，所求出的 I，J 值则是该元件左（或上）端接点的 I，J；于是，由此 (I, J) 接点记录的右（或下）联元件指针域便可检索到被击中的元件记录。此外，下述算法步骤只阐述了主要思路，具体应用时还需要作细致的考虑并作适当修改。

(1) I-J 算法

$A\cdot1$ $x^* - x_0 \Rightarrow \langle x\rangle$；$y^* - y_0 \Rightarrow \langle y\rangle$。

$A\cdot2$ $\left[\frac{y}{d}\right] \Rightarrow \langle I\rangle$；$y - I * d \Rightarrow \langle y\rangle$；

$\left[\frac{x}{d}\right] \Rightarrow \langle J\rangle$；$x - J * d \Rightarrow \langle x\rangle$。

$A\cdot3$ $I + I_0 \Rightarrow \langle I\rangle$；$J + J_0 \Rightarrow \langle J\rangle$。

$A\cdot4$ 追踪接点链，找出 (I, J) 接点（或器件记号）记录；此记录始址$\Rightarrow DN1$。

$B\cdot1$ $y \leqslant \mathrm{Max}(b, d')$？否，则转 $C1$。

$B\cdot2$ $x \leqslant \mathrm{Max}(b, d')$？否，则转 $D1$。

$B\cdot3$ 退出。（击中对象为 (I, J) 接点或器件记号）

$C\cdot1$ 找出 $(I+1, J)$ 接点记录，记录始址$\Rightarrow DN2$ 备用。

$C\cdot2$ $'4'\Rightarrow UDLR1$；$'2'\Rightarrow UDLR2$ 备用。

$C\cdot3$ $((DN1)+4)\Rightarrow DEL$。

$C\cdot4$ 退出。（击中对象为垂直元件，其记录始址在 DEL 单元中；其下端节点记录始址在 $DN2$ 单元中）

$D\cdot1$ 找出 $(I, J+1)$ 接点记录，记录始址$\Rightarrow DN2$ 备用。（右端接点）

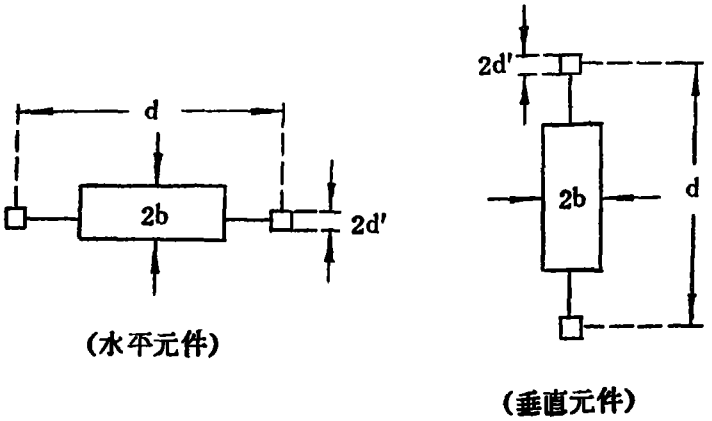
$D\cdot2$ $'5'\Rightarrow UDLR1$；$'3'\Rightarrow UDLR2$ 备用。

$D\cdot3$ $((DN1)+5)\Rightarrow DEL$。

$D\cdot4$ 退出。（击中对象为水平元件，其记录始址在DEL中）。

(2) 说明

上述算法中，d为接点间距，$2b$为各种元件图的最大宽度，$2d'$为接点或器件记号中最大尺寸（见图6）。对于记录始址为（DEL）的元件，其左（或上）联接点的记录始址为（$DN1$），其右（或下）联接点的记录始址则为（$DN2$）。由图2还可看出：（$DN1$）+（$UDLR1$）及（$DN2$）+（$UDLR2$）均是指向（DEL）的指针域。



（水平元件）　　（垂直元件）

图 6

(3) 消去一个元件的算法

$A\cdot1$ 元件记录回收入自由表；从同类元件链中解结。

$B\cdot1$ 元件的左（或上）接点记录的引线域数减1，即（（$DN1$）+1·引线域）－1 ⇒（$DN1$）+1·引线域。

$B\cdot2$ （（$DN1$）+1·引线域）=0？是，则转$B\cdot4$。

$B\cdot3$ 0⇒（$DN1$）+（$UDLR1$），转$C\cdot1$。

$B\cdot4$ 回收始址为（$DN1$）的接点记录；从接点链解结。

$C\cdot1$ （（$DN2$）+1·引线域）－1⇒（$DN2$）+1·引线域。

$C\cdot2$ （（$DN2$）+1·引线域）=0？是，则转$C\cdot4$。

$C\cdot3$ 0⇒（$DN2$）+（$UDLR2$），转$D\cdot1$。

$C\cdot4$ 回收（$DN2$）记录；从接点链解结。

$D\cdot1$ 使被消元件（可能还有一或两个端接点）的显示档案隐去。

关于显示档案的隐去可用一条跳转指令实现。在缓存中此档案并未取消，但在下一次按数据结构重构电路图时，则被消部分就不会再占缓存了。

消去一个接点时，它所联接的元件均应消去，因而其算法较上述要复杂，此处不多论列。

参 考 文 献

[1] E. Wolfendale, Computer-Aided Design Techniques (Ch.2), Butterworths, London, 1970.

[2] 罗伯鹏，光笔图形显示机在计算机辅助设计中的应用，阳朔 CAD 会议资料，1978.10.

[3] 易晓东译，对话式图形显示程序剖析：用于存贮管显示终端的电路，《专题技术译丛》第 18 期，国防科技大学，1979.4.

An Approach to the Construction of Circuit Graph and the L.P.Searching in the Interactive CACD System

Luo Bo—peng

Abstract

On the basis of the displayed joint's row-column information stored in the data structure, this paper present an algorithm for the regeneration or translation or scaling-up of the displaying circuit graph and another for the searching and correcting of the record relevant to an object hit by the light pen.