マイクロコンピュータとワークステーション50 － 4
（１９８８．５．１３）

アイコニック言語Ｄｉａｌｏｇ．ｉによる ＣＡＤ用マン-マシン インタフェース

田代　秀一　，　岡田　義邦
電子技術総合研究所

複雑な概念を図的に表現して計算機と対話する手段として，様々なアイコニック言語が提案されている．アイコニック言語はその性質上汎用性を持たせることは困難であり，一方応用毎に異なった言語を開発するとそのコストは大きい．そこで多様なアイコニック言語を構成する基盤となるシステムが必要と考えられる．本稿では，このような基盤システムであるメタなアイコニック言語Dialog.iの提案を行ない，その上に論理ＣＡＤを構成する例について紹介する．

Iconic language Dialog.i and its man-machine interface as CAD system.

Shuichi Tashiro and Yoshikuni Okada
Electrotechnical Laboratory
1-1-4, Umezono, tsukuba, Ibaraki, 305 Japan

Dialog.i is a meta-iconic language which supports basic environment for developping various iconic languages for various apprications. You can define an icon as an object which contains a definition of pictograph, message ports on the pictograph and methods to process messages arrived on the ports. We discuss the summary of Dialog.i and its application to logic circuit CAD system.

1．はじめに

計算機と人間との対話に図的表現を用いることが適当であるような場面は多い．ＣＡＤ，プロセス制御，オフィスオートメーション，ホームオートメーション，各種エキスパートシステム等がその例である．

利用者の意志をシステムへ伝える仲立ちとして図形を用いる試みは，既に各所で行なわれ，そのいくつかは実用化されている．

最も基本的な方式は，XEROXのstarあるいはアップルマッキントッシュ等に見られるように，あらかじめ定義されたアイコンをディスプレイに表示しておき，利用者がそのアイコンに対し，例えばマウスポインタを合せてボタンを押すといった‘操作’を加えることによるものである．この方式は，単純な操作を１回１回システムに伝える為には便利であるが，定型的な操作の繰返し，さらにそれに条件判断が加わったもの等，より複雑な概念をひとつのプログラムとして伝えるといったことはできない．

これを発展させ，アイコンの２次元的な配置あるいはアイコンの結合状態によってプログラムを記述し，実行させようと言う試みもある[1]-[3]．アイコンは，通常の文字を使った１次元的言語における単語に相当し，アイコンの配置および結合に対して意味付けを与えることが文法に相当する．このような言語は一般にアイコニック言語と呼ばれる．

しかし，文字による１次元的言語のように文法の厳密に定義された一つのアイコニック言語を，最初に述べたような様々な応用に適用することは困難と考えられる．

なぜなら言語の汎用性を高めようとすると文法事項即ちアイコン間の空間的関係に持たせる意味を極めて単純で一般的なものとせざるを得ず，その結果，対象世界を計算機上に自然な形で写し取ることが困難となってくるからである．文字による言語の文法要素の一部即ち関数呼出あるいは制御の流れ等を視覚化しただけでは多少視認性の向上があっても，表現された内容全体の理解の助けには不十分であろう．

より良いマンマシンインタフェースを確保するためには，応用毎にそれに適したアイコニック言語を開発しなくてはならないことになる．今後，アイコニック言語の開発を支援するシステムを整えることが重要となってくるだろう．

そこで，各種応用向けアイコニック言語を構築する基礎となる，メタなアイコニック言語であるＤｉａｌｏｇ．ｉの開発を開始した．

Ｄｉａｌｏｇ．ｉの上に構成する応用向け言語としては，先ず論理ＣＡＤを１つの目標としている．

2．システムへの要請

アイコニック言語は，システムと人間が視覚的情報を仲立ちとして会話するための手段である．人間にとって理解し易く，又操作し易い言語仕様を設計することが重要となる．図を使った全く新しい抽象的な言語を設計するといったアプローチも考えられるが，対象となる世界を即物的に，自然な形でディスプレイ上に写し取ることのできる表現力を持った言語の必要性は大きいであろう．

たとえば論理回路を記述しようとする場合，論理式を図的表現に置き換えるような言語を作るのではなく，ロジックシンボル，配線，回路ブロックなどを直接にアイコンに表現し，そのアイコンを，あたかも紙の上で製図を行なうように配置し，配線して一つの回路を設計してゆける環境を整備する必要があろう．この場合，論理回路を描くことが即ちアイコニック言語を用いたプログラミングとなる．このようなアイコニック言語を作成すると，それはそのままＣＡＤとして利用できる．

こうしてできたプログラムを‘実行’することにより，実際の回路の動作をシミュレートする，あるいは，回路の接続状態をネットリスト等の記号表現の形で取り出し，ＬＳＩの自動設計等，後続の処理に利用するなどができる．

プロセス制御では反応器，配管，状態の表示等を，ホームオートメーションでは家の見取図，各種電気製品，操作メニュー等をアイコンに代表させることになろう．

アイコンはきわめて様々な対象を表現できる必要がある．また，実世界の物体が様々な相互作用を及ぼし合うように，アイコンも互に相互作用を及ぼし合える必要がある．又，人間が容易にアイコンに対して操作を加えることのできるようなインタフェースも必要である．

このようなアイコンを構築する基盤となるよう，次の様な要件を満たすシステムの開発を目指す．

1)アイコンをオブジェクトとするオブジェクト指向言語である．

アイコンの定義の中には，図的定義即ちそのアイコンがディスプレイ上にどのような形を表現するかの定義とメソッド即ちそのアイコンの持つ意味を手続的に定義したものが含まれる．

2)アイコンに対する人間の操作（例えばそのアイコン上のある位置へマウスポインタを合せてボタンを押す，あるいはボタンを押しながらマウスを移動する等）を，メッセージの形でそのアイコンへ伝達する手段を持つ．

3)アイコンの基本的な位置関係を把握し，アイコンからの要求に応じてレポートする機能を持つ．

4)アイコンを階層的に定義でき，様々な角度からその定義状況を表示することが出来る．

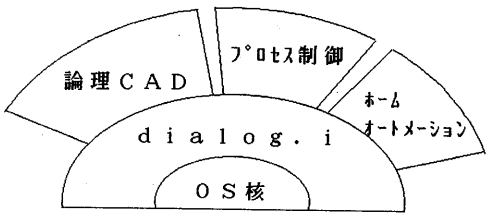


図１　Ｄｉａｌｏｇ．ｉの位置づけ

この様な基本システムを先ず実現し，その上に応用に適した様々なアイコンを定義することによって，応用向けのアイコニック言語を実現して行くことがシステム開発にとって効率的であると考えている．応用向けのアイコニック言語の文法は，各アイコン自体の中に分散的に定義されることになる．

システムの位置付けを図１に示す．

３．Ｄｉａｌｏｇ．ｉの概要

Ｄｉａｌｏｇ．ｉはその上に様々な仕様のアイコニック言語を構成するためのメタなアイコニック言語である．

Dialog.iにおけるアイコンは，図２のように図形とメソッドの定義を備え，外部からメッセージを受けとることによってメソッドが起動されるオブジェクトである．

3.1 アイコン図形の定義

図形の定義法は，Dialog.iに用意されたグラフィックエディタを用いて，ビットマップ上に描く固定的なものと，メソッドの定義の一部として，プログラムによって定義する可変的なものの２種類を用意する．

これらの組合せで図形を定義することもできる．図形はアイコン内部のローカルな座標で定義する．

プログラムによる定義のために，Dialog.i

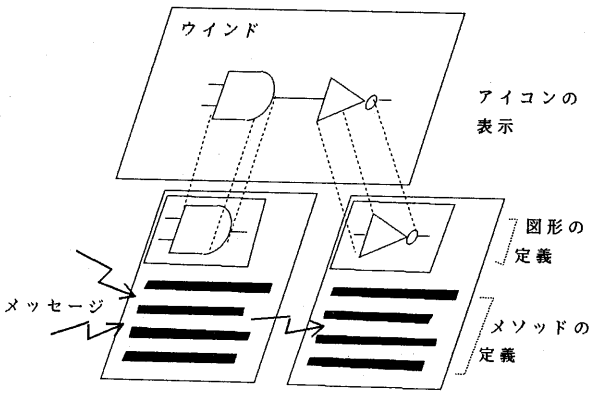


図２　アイコンの定義

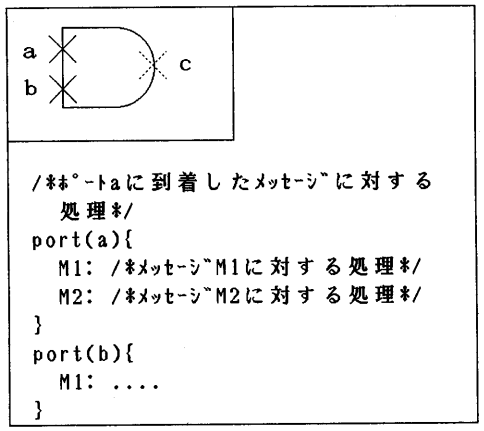


図３　ポートの定義
（×はポート座標の定義を示す）

は点，直線，円等を描くための基本的な描画命令を提供する．

3.2 ポート

アイコンへのメッセージはアイコン内に定義されたポートに対して送られる．

ポートは，図形上の任意の座標点あるいは座標範囲に任意の個数設定することができる．

ポートには，receive，send，send/receiveの３種類がある．

receiveポートはメッセージの受け口であり，一つのポートに対して複数のメソッドを定義することができる．ポートとそれに対応したメソッドの定義の例を図３に示す．

ポートには図の座標で定義するポートの他に，図と対応付けずに名前だけで指定する物，およびdefaultポートがある．

sendポートはメッセージの送り口である．メソッドの中で，sendポートの名前に対してメッセージを送る手続をすると，そのメッセージはsendポートに対応してシステムに設けられたキューに一旦蓄えられ，そのsendポートに空間的に接触しているreceiveポートが存在している場合そちらへ送られる．メッセージの実際の送り先の決定は，後に述べる空間マネージャによって行なわれる．

send/receiveポートは上記２つの機能を兼ね備えたものである．

3.3 メソッドの記述

アイコンがメッセージを受取った場合に起動される処理は，Ｃ言語に類似した言語で記述する．

3.4 アイコンの階層的記述

幾つかのアイコンを結合した集団に対してそれを代表する図形を与え，一つのアイコンとして再定義することができる（図４参照）．

3.5 利用者とｱｲｺﾝとのｲﾝﾀﾌｪｰｽ

Dialog.iが直接にサポートする利用者-アイコン間インタフェースは，マウスボタンの押された事およびボタンの押された状態でマウスが移動したことを示すメッセージを，その時マウスポインタの下にあるアイコンに対して送ることだけである．

メッセージは通常，アイコンのdefaultポートに対して送られる．アイコン図形上にポートが定義され，マウスポインタがそのポート上に位置している場合は，メッセージはそのポートに対して送られる．

メッセージの送り先となるのアイコンの選択，その中のポートの選択は，後に述べる空間マネージャによって行なわれる．

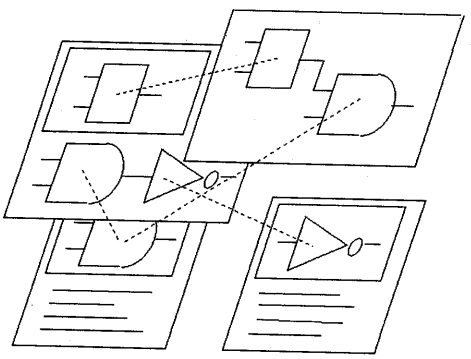
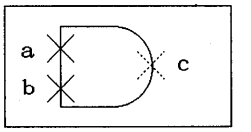


図４　アイコンの階層的定義

利用者がアイコンに対して，より複雑なメッセージを送る必要のある場合は，例えばメニューを表示して利用者に送りたいメッセージを選択させる機能を持ったアイコンを定義し，そのアイコンを通して目的のアイコンにメッセージを送るようにアプリケーションを構成すれば良い．又，Dialog.iでは，アイコンが動的に他のアイコンを生成することが可能なので，この機能を用い，例えばあるアイコンの，あるポート上でマウスボタンが押されるとメニューアイコンが自動的に起動，表示されるようにアプリケーションを作ることができる．




```
port(c){
   mouse_crick:
      create_icon(line);
}
port(a){
   sig(x):send_port(c,sig,x & y);
}
port(b){
   sig(y):send_port(c,sig,x & y);
}
```

図5 ロジックシンボルの定義例

3.6 空間マネージャ

空間マネージャは，ある平面上に存在するアイコンおよびその中に定義されているポートの座標を把握し，管理する．

空間マネージャの機能には以下のものがある．

(1)アイコンのsendポートとreceiveポートの接続関係を把握，管理し，sendポートから出力されたメッセージを，そこに接続しているreceiveポートへ配達する．

(2)マウスポインタの座標を把握し，マウスボタンが押されると，そのことを伝えるメッセージを生成してマウスポインタの下に位置するアイコンへ送る．

(3)アイコンの移動中に他のアイコンに接触したことを検知すると，そのことを知らせるメッセージを，接触した双方のアイコンのdefaultポートへ送る．

(4)アイコンからの座標情報の問い合せに答える．

4．論理ＣＡＤへの適用

図5に，Dialog.iの上に論理ＣＡＤを構築する際のアイコンの定義の一例を示す．

ロジックシンボルのアイコンの出力ポートに，マウスボタンが押されたというメッセージを受けると結線アイコンを生成するようにメソッドを記述するとともに，結線アイコンの図形を動的に定義してその先端がマウスの移動先に向って伸びて行くようにプログラムしておけば，利用者は，ロジックシンボルとロジックシンボルの間に，あたかもマウスを用いて線を描いて行くような感覚で配線の定義を行なうことが出来る．空間マネージャからの接触メッセージを利用すれば，他のアイコンに重ならないように制御しながら配線を伸ばして行くような制御が可能である．

入力ポートに論理値のメッセージを受けると，それに対してある論理演算を施した結果を出力ポートへ出力するようなメソッドを定義すれば，でき上がった図は直ちに論理シミュレータとして機能させられる．

また，配線済みの回路の接続情報を記号表現で出力したい場合，各ロジックシンボルのアイコンに対応する論理式を定義しておき，入力ポートが他のアイコンの出力ポートと接続されたというメッセージを受取ると，自己の定義式の中のパラメータを，接続によって同値となる他のアイコンのパラメータで置換して行くようなメソッドを定義すれば良い．配線終了後に各アイコンからこの論理式を集めてくれば，全体の論理式を完成することができる．

5．CADとしてのﾏﾝｰﾏｼﾝｲﾝﾀﾌｪｰｽ

5.1 構文エディタ機能

プログラミングのためにアイコンを配置して行く操作を，アイコンに定義された様々な空間的制限条項に沿ってガイドしてゆく機能即ち文字による言語における構文エディタに相当する機能を充実させれば，そのアイコニック言語は，エンドユーザにとって使い良いＣＡＤとして機能する．

Dialog.i自体は，アイコンを配置する為の原始的エディタを持つだけで，構文エディタ機能は持たない．しかし，４．で述べた結線アイコンの例のように，アイコン自体の中に自分がどう振舞うべきかを定義しておくことにより，構文エディタ機能を実現できると考えている．構文エディタ機能を各アイコンの中に分散配置することは，アイコンの表現する対象が複雑で，アイコン毎に空間的振舞の規則が異なっているような場合に有利である．また，アイコンの種類を後から増やしてゆくために有利であり，システムの拡張性を大きくすることができる．

5.2 表示機能

作成途中又は完成した図面を利用者に対して見やすく提示する機能が要求される．

Dialog.iは，階層的に定義されたアイコンの結合状態を，階層毎にウインドを割り当て，マルチウインドで表示する機能を持つ．

現在注目していないウインドは表示せずに隠してしまう．隠れているウインドは，それを使用して定義された上位アイコンをマウスポインタで指示し，ウインド呼出用のボタンを押すことによって呼出す．この機能により，利用者は注目箇所を次々とズームアップして行く感覚で閲覧して行くことが出来る．

図面が全て計算機内部に記憶されており，利用者が高解像度ディスプレイを自由に利用できる環境にある場合，利用者の注目している箇所を素速く，ダイナミックに表示することに重点を置いた表示方式が重要となろう．これに比較して，定義されたアイコンの全てを２次元的に美しく配置した大きな図面を描くことの要求される頻度はさほど大きくないと考えている．したがってこの様なプリティプリンタ機能はDialog.i内部には実装せず，必要に応じてアプリケーションとして実現するよう考えている．

階層的な設計は，しばしば人間の自然な発想の順序と対立する場合がある．思いついた部分をまず書き留め，後から，既に書き留めてある部分との対応付けと結合を行なって全体を構成して行くといった設計手順がその例である．

このような設計手順をサポートするために，Dialog.iでは，同一アイコンの重複利用，独立に定義されたウインドの結合の機能を持たせる予定である．

6．おわりに

本稿では，図の意味，図に対する操作，図と図の関係等を記述するための汎用的な言語であり，それによって記述された対象を実行するための基本処理系であるDialog.iの概要を述べ，その上に構成する応用向けアイコニック言語の例として論理ＣＡＤの構成例を紹介した．Dialog.iは，現在Sunワークステーション上に開発中である．

このような基本システムが実現すれば，その上に用途に応じ様々な仕様のアイコニック言語を容易に記述できるようになり，今後ますます需要の高まる高度なマン-マシンインタフェースの実現に貢献できると考えている．

【文献】

[1]Shi-Kuo Chang,"Visual Languases : A Tutorial and Survey", IEEE Software, Vol. 4, No. 1 , pp29-39 1987.
[2]I. Yoshimoto, N. Monden, M. Hirakawa, M. Tanaka, and T. Ichikawa,"Interactive Iconic Programming Facility in HI-VISUAL", 1986 IEEE Comp. Soc. Work. on Visual Languages, pp. 34-41.
[3]F. Lakin,"Visual Grammars for Visual Languages", AAAI '87 pp. 683-688