# A Computer-Aided Circuit Layout System Based on the Functional Structure and the Physical Structure of Circuits

By

Takao OZAWA* and Hidenori SEKIGUCHI**

(Received March 28, 1983)

**Abstract**

In this paper, there is reported a computer-aided circuit layout system which is based on a new design philosophy. For a circuit to be designed and laid out, two structures called the functional structure (FS) and the physical structure (PS) are defined. The former is concerned with the behavior of the circuit and is hierarchical, while the latter is dependent on the physical realization of the circuit. The FS has "blocks," "components," "connections, "terminals" and "external terminals" as its basic elements, whereas the PS has "modules," "nets," "pins" and "external pins" as its basic elements. A circuit description language to specify the FS is designed and its interpreter is computer programmed. In order to have access to specific basic elements, data reference formulae are defined. The realizations of the basic elements of the PS are displayed on a color CRT. Our circuit layout procedure consists of three steps: (1) inputting the FS by using the circuit description language, (2) assigning components and terminals to modules and pins respectively, and (3) layouting basic elements of the PS on the board by using layout and display commands which take data reference formulae as their operand. The last step is performed conversationally and the layout of the elements can easily be changed.

## 1. Introduction

Many computer-aided IC (integrated circuit) layout systems have been developed as the sizes of IC's become larger and the manual circuit layout becomes more difficult because of the increased complexity of the circuits.[1] Among them are computer-aided systems introducing a hierarchical circuit design and layout.[2)-5)] In these systems an IC is designed through hierarchical steps: it is first constructed as a system of interconnected sub-circuits having certain specified macro functions. Then, the sub-circuits are constructed as collections of sub-sub-circuits, and so on until certain elemental

* Department of Electrical Engineering II.

** A former graduate student at the Department of Electrical Engineering II. Now with Fujitsu Ltd., Kawasaki.

circuits are implemented. The circuit layout is usually considered after or at the later stages of the circuit design. Unfortunately, the circuit design which ts to achieve the desired overall circuit function and the circuit layout which is to place the circuit on a semiconductor chip or a printed circuit board, are often inconsistent. The former is more concerned with the behaviour of the circuit and is more or less software oriented, while the latter is very much dependent on the physical realization of the circuit, and thus is hardware oriented. (This inconsistency will become clearer when the reader proceeds to sections 2 and 3.) The previous computer-aided systems seem to ignore this inconsistency and try to use one hierarchical structure for both the design and layout. Therefore, inconveniences arise when the designer wants to approach the laid out circuit with the knowledge on the circuit design. He has to have a full understanding of the relation between the circuit functions and the laid out circuit, which becomes extremely difficult when the circuit size becomes large.

In this paper, we present a computer-aided circuit design and layout system based on a new philosophy concerning the structure of the circuit. The structure which can be defined from the functions of sub-circuits (and sub-sub-circuits and so on), and the structure which can be defined from the physical realization of the circuit are clearly distinguished. The former is called the functional structure (FS), whereas the latter, the physical structure (PS), and a language and a data base for describing the two structure are designed and computer programmed. Our system is mainly aimed to layout a hybrid IC, which consists of circuit modules (components) and electrically conductive wires laid out on a board. However, the system can also be used to layout other types of integrated circuits.

## 2. Two Structures of A Circuit

### A. Functional Structure (FS)

As stated in the previous section the process of the circuit design is hierarchical. The FS is defined in accordance with the design process and is represented by a rooted tree, such as shown in Fig. 2. 1. The tree in the figure indicates that the circuit A contains B1, B2 and B3 as its sub-circuits; sub-circuit B1 contains C1 and C2 as its sub-circuits, and so on. The hierarchy of the structure is indexed by "levels" in the rooted tree. In Fig. 2. 1 circuit A is at level zero, the highest level, sub-circuits B1, B2 and B3 are at level one, sub-sub-circuits C1, C2 and C3 are at level two, and so on. Levels of leaves of the rooted tree are called leaf levels. (We assume there are always more than one level, and thus the root cannot be a leaf at the same time.) In our structural circuit description, the functional units at each level, except those at a leaf level, are called "blocks," and those at a leaf level, "components." If a block at a level contains blocks at one level below. (Level two is one level below level one and so on.) the latter blocks are called "components" of the former. Therefore B1,

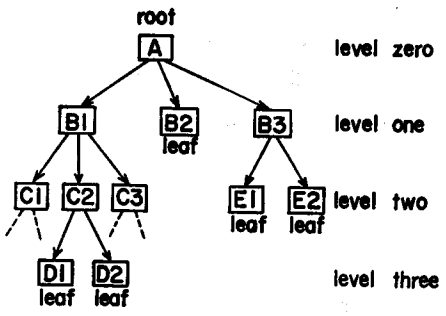


Fig. 2. 1. An Example of the rooted tree representing the Functional Structure.

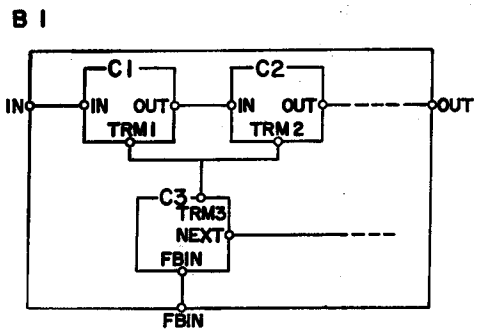


Fig. 2. 2. An example of a block. Block: Bl. Components: C1, C2 and C3. Terminals: IN, OUT and TRM1 of C1, IN, OUT and TRM2 of C2, and so on. External terminals: IN, OUT and FBIN. Connections: (IN, IN of C1), (TRM1, TRM2, TRM3) and so on.

B2 and B3 are components of A, and C1, C2 and C3 are components of B1. B2, D1, D2, E1 and E2 are components at leaf levels.

Components in a block are, in general, electrically inter-connected. This is represented by "connections" which connect "terminals" attached to components. There are also electrical connections from components within a block to those outside the block or to external circuits if the block is at the highest level. In order to achieve such electrical connections a block is provided with "external terminals." In our system, a connection is specified by a set of terminals and/or external terminals which it connects. An example of the finer structure in a block is illustrated in Fig. 2. 2. Obviously, an external terminal of a block is a terminal, if the block is identified to be a component contained in a block at one level higher.

The "blocks," "components," "terminals," and "external terminals" are called the basic elements of the FS.

### B. Physical Structure (PS)

The physical realization of a circuit varies depending on the form of the circuit integration. In general, however, it consists of elemental sub-circuits specified by the manufacturing process and wires connecting them. In our system, the elemental units of circuit realization are called "modules." A module is equipped with "pins" to provide electrical inter-connections with other modules. The circuit has "external pins" to provide electrical inter-connections with external circuits. Wires connecting electrically equi-potential pins and/or external pins are called "nets." In our system, a net is specified by the pins and/or external pins which it connects. "Modules," "pins," "external pins" and "nets" are the basic elements of the PS of a circuit. Note that the PS is not hierarchical.

**C. Relation Between the Functional Structure and the Physical Structure**

The relation between the FS and the PS is schematically illustrated in Fig. 2. 3. A component at a leaf level of the FS only has a corresponding physical realization, that is, it is realized as a module or as a part of a module in the PS. (A module may contain more than one elemental circuit sush as a gate or a fiip-fiop.) The terminals of such a component at a leaf level correspond to the pins of the module. Also, the external terminals of the block at the highest level of the FS correspond to the external pins of the PS. In general, the rest of the terminals in the FS have no correspondence in the PS. The connections in the FS are realized as nets in the PS. A net, however, has a finer structure, as will be described in Section 4, which is not directly related to the connections.

Based on the PS, the laid out circuit is displayed on a color CRT. It can be approached by the circuit designer with the knowledge of the FS.

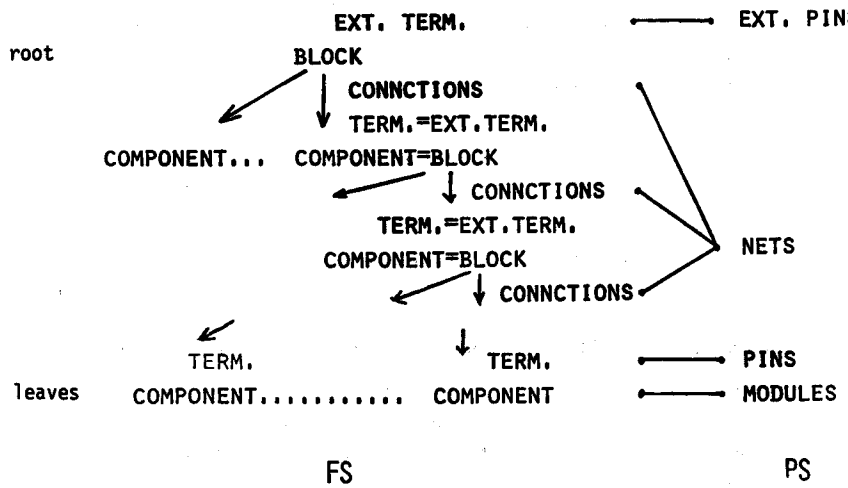


Fig. 2. 3. Relation between the FS and the PS.

## 3. Circuit Description Language (CDL)

For the input of a circuit into the system, a circuit description language together with its associated interpreter has been developed. This language can describe the FS of the circuit, and has the following features:

1. Sub-circuits can be input block by block.
2. Repeated input of sub-circuits having the same sub-structure can be done by simple operations.
3. Basic structural elements can be labeled easily by use of group expression.
4. Input errors are checked. Especially the consistency regarding the FS is tested automatically.

Table 3. 1 Formal Specification of Circuit Description Language

(a) basic symbols

[alphabet] ::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
[digit] ::=0|1|2|3|4|5|6|7|8|9
[name] ::=[alphabet]|[name] [alphabet]|[name] [digit]
[number] ::=[digit]|[number] [digit]
[interval] ::=[number]|[number]:[number]
[identifier] ::=[name]|[number]|[name]-[number]
[inentifier group] ::=[name]|[interval]|[name]-[interval]

(b) circuit description

[circuit description] ::=[block description]|[circuit description] [block description]
[block description] ::=[block division] [external division] [component division] [connection division]|[block division] [component division] [connection division]|[block division]

(c) block division

[block division] ::=BLOCK;|BLOCKb [qualified identifier];|BLOCKb [qualified identifier] b[block attribute];
[block attribute] ::=SAME([qualified identifier])|LIKE([qualified identifier])
[qualified identifier] ::=[identifier]|[qualifier] [identifier]
[qualifier] ::=[identifier].|.|[qualifier] [identifier].

(d) external division

[external division] ::=EXTERNALb([identifier group row]);
[identifier group row] ::=[identifier group]|[identifier group row]b[identifier group]

(e) component division

[component division] ::=COMPONENTb [component declaration row];
[component declaration row] ::=[component declaration]|[component declaration row], [component declaration]
[component declaration] ::=[identifier group] ([identifier group row])|[identifier group]

(f) connection division

[connection division] ::=CONNCTIONb [connection declaration row];
[connection declaration row] ::=[connection declaration]|[connection declaration row], [connection declration]
[connection declaration] ::=[identifier group] ([terminal identifier row])|([terminal identifier row])
[terminal identifier row] ::=[terminal identifier]|[terminal identifier row]b[terminal identifier]
[terminal identifier] ::=[identifier group]*[identifier group]|[identifier group]*[identifier group]

Note: The parts of keywords not underlined can be omitted. b: blank.

A formal specification of our circuit description language is given in Table 3. 1. The input of a circuit into the system must be in accordance with the hierarchy of the FS, that is, the block at the highest level must be input first, and a block at a lower level can be input only after all its ancestor blocks have been given to the system.

(i) identifiers

A basic structural element is labeled by an "identifier" to distinguish it from others.

An identifier is a "name," a "number" or a name with a number as its "subscript." Within a block different identifiers must be given to different components, terminals or connections respectively, but a terminal, for instance, can be given the same identifier as a component. A series of numbers can be specified as an "interval," and thus a group of identifiers can be given using an interval. For example, A-1 : 3 is equal to a group of identifiers A-1, A-2, A-3. An identifier of a terminal, as a rule, follows the identifier of the component to which it is attached, and an asterisk(*) is placed between the two identifiers. If no particular terminal of a component need be specified, only an asterisk following the identifier of the component, such as C1*, is input. An external terminal is specified by an identifier following an asterisk, for example *EX.

In order to express the relation between a block and a component or a terminal within it, a "qualified identifier" is introduced. This is a component or terminal identifier following a "qualifier," which is the identifier of the block followed by a period(.). For blocks at level one or below, qualifiers can be repeated. For example, components X, Y and Z in a level-2 block N which in turn belongs to a level-1 block A are specified by A. N. X, A. N. Y and A. N. Z respectively. The three qualified identifiers as a group can be specified by A. N. X. Y. Z.

(ii) block division

A block division initiates a block description. In the case of the highest level block this consists of the keyword "BLOCK" and ";" and in the case of a lower level block, of BLOCK and the qualified identifier of the block. If a block to be input has the same structure as a block already described, the qualified identifier of the latter is given as the "block attribute" of the former, and then the description of the components, terminals and connctions of the former can be omitted. The block attribute "SAME" indicates that the entire hierarchical structures of the two blocks are the same, and

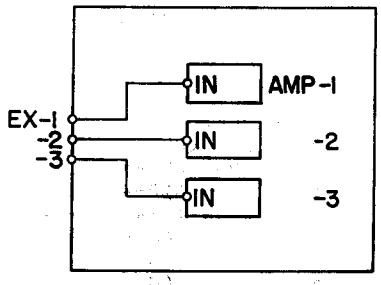


Fig. 3. 1. Group expression of connections.
(*EX-1 : 3 AMP-1 : 3*IN) = (*EX-1 AMP-1*IN), (*EX-2 AMP-2*IN), (*EX-3 AMP-3*IN).

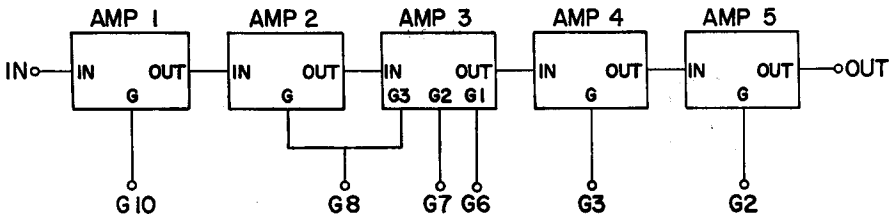


Fig. 3. 2. Block diagram of a hybrid IC

the block attribute "LIKE" indicates that the two have the same structure of one level only. (The components in the blocks may have different finer structures.)

(iii) external division

An external division declares external terminals. This division is necessary for the highest level block only. A lower level block is a component contained in another block at the same time, and thus its external terminals need not be given.

(iv) component division

A component division declares all the components and terminals belonging to the block. If the terminal identifiers of more than one component are the same, they can be given as a group. For example, COMP-1: 2(IN OUT) is equal to COMP-1(IN OUT), COMP-2 which is equal to COMP-1(IN OUT), COMP-2(IN OUT) where IN and OUT are terminal identifiers, and each of components

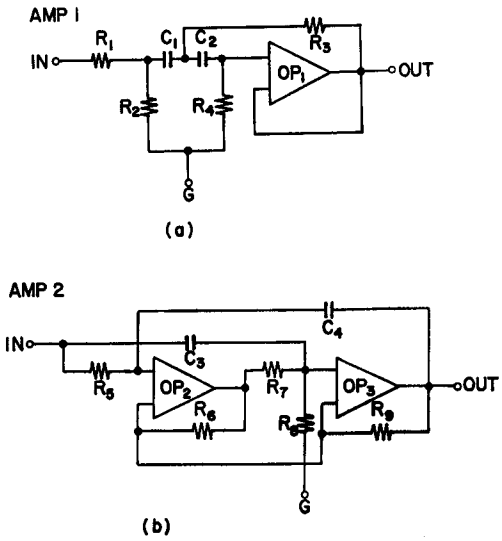


Fig. 3. 3. Circuit diagrams of amplifiers

```
 1  BLOCK;
 2    EXT (OUT G-2:3 VM-5 G-6:8 VM-9 G-10 IN VP-12);
 3    COMP AMP-1:2(IN OUT G),AMP-3(IN OUT G-1:3),
 4         AMP-4:5(IN OUT G),POWER-1:4(VP VM);
 5    CONN (*IN AMP-1*IN),(AMP-5*OUT *OUT),
 6         (AMP-1:4*OUT AMP-2:5*IN),
 7         (AMP-1*G *G-10),(AMP-2*G AMP-3*G-3 *G-8),
 8         (AMP-3*G-1:2 *G-6:7),(AMP-4:5*G *G-3:2),
 9         (*VP-12 POWER-1*VP POWER-2*VP POWER-3*VP POWER-4*VP),
10         (*VM-5 POWER-4*VM POWER-3*VM),
11         (*VM-9 POWER-2*VM POWER-1*VM);
12  BLOCK AMP-1;
13    COMP OP-1(1:3),R-1:4(1:2),C-1:2(1:2);
14    CONN (*IN R-1),(R-1 R-2 C-1),(R-2 R-4 *G),
15         (R-4 C-2 OP-1*3),(C-1 C-2 R-3),
16         (OP-1*1 OP-1*2 R-3 *OUT);
17  BLOCK AMP-2;
18    COMP OP-2:3(1:3),R-5:9(1:2),C-3:4(1:2);
19    CONN (*IN R-5 C-3),(R-5 OP-2*3 C-4),(R-8 *G),
20         (OP-2*2 OP-3*2 R-6 R-9),(OP-2*1 R-6 R-7),
21         (C-3 R-7 R-8 OP-3*3),(OP-3*1 R-9 C-4 *OUT);
```

Fig. 3. 4. Circuit description of the circuit shown in Figs. 3. 2. and 3. 3.

COMP-1 and COMP-2 has terminals IN and OUT. Terminals can be also declared as a group. For example, IC (1:8) is equal to IC (1 2 3 4 5 6 7 8), which can also be given as IC (1:4 5:8).

(v) connection division

A connection division declares connections, each of which is described by a set of terminal identifiers. It is not necessary to give an identifier to a connection. An example of connections and their description are given in Fig. 3. 1.

Example

The circuit shown in Figs. 3. 2 and 3. 3 can be inputted by the circuit description given in Fig. 3. 4. (The description of AMP-3 to AMP-5 is omitted.)

## 4. Realization of the Physical Structure

The basic elements in the PS have their physical realizations laid out on a board. The board may have more than one "layer." Then modules and their pins can be placed on any of the layers, and nets can go from one layer to another going through "through holes." The realization of a module is represented by a polygon, which is specified by a set of "x-y coordinates," that is, the positions of vertices of the polygon on the board. The realization of a net is represented by a "rooted tree," and the pins it connects, by "nodes" of the rooted tree. A node of the rooted tree may also represent a through hole or a "branching point," which is a point where three wires meet. A branch of a rooted tree represents a wire connecting two of the pins, through holes and/or branching points. Now such a wire may not be straight. Then it is a collection of straight wiresegments. A new point (or points) called a "turning point" where two straight wire-segments meet, is introduced. (A turning point is not given as a node of the rooted tree.) Thus, in the PS, a net is specified by a set of pins, but for its realization on the board the pins, through holes, branching points and/or turning points together with their positions (x-y coordinates) need be specified. The number of branches at a node, that is, the degree of a node is limited to three or less from a practical point of view. An example of a rooted tree is shown in Fig. 4. 1(e), where the pin which is specified as the root of the tree is indicated by a black node and the remaining pins, by white nodes.

In the routing process a net cannot, in general, be completed at one step. First a single wire of the net is routed, and next unconnected wires are added here and there, which are then tied up to form a net. In Fig. 4. 1, a net connecting five pins is completed by the process as shown in the figure. Before a net is completed it is represented by more than one rooted tree. When two rooted trees are merged to one by the addition of a branch, one of the roots is eliminated. Besides, the directions of the branches belonging to the tree whose root has been eliminated may have to

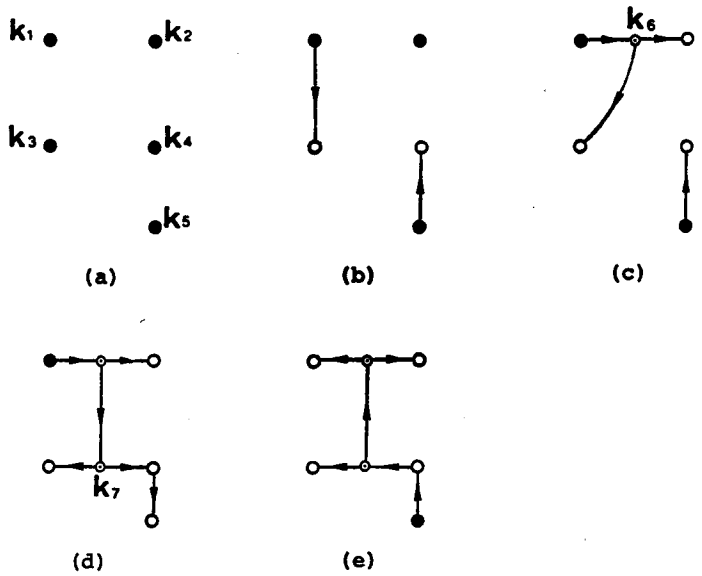


Fig. 4. 1. Construction of a rooted tree representing a net.

be changed.

The elements of the PS laid out on a board can be displayed on a color CRT. The designer may want to approach an element on the CRT. This he can do by using the graphic cursor equipped with the CRT, that is, if he points out a pin by the cursor, the system finds out its identifier from the position of the pin which is equal to that of the cursor. To identify a module or a net he must approach one of its pins first. To make the above searching process efficient, the board is divided into "pages," and all the pins in a page are linked to form a list. The page containing the specified pin is first found from the position of the cursor, and then the pin is sought on the list, thus minimizing the number of pins to be songht.

## 5. Data Base

The data base of the system reflects the two structure of the circuit, and its main part consisis of data structures for the basic elements of the FS and PS. For the FS, there are provided six "entities" which are associated with blocks, components, connections, terminals and names and subscripts, respectively. The entities for the PS are associated with modules, nets, pins, x-y coordinates, roots, through holes and pages respectively. The entities are given in Table 5. 1. As shown in Table 5. 1, each entity has a descriptor and/or a directory. The location of a descriptor in the computer memory is called a "code," and a descriptor can be approached by specifying its code. Codes are generated internally by the system.

Table 5. 1 Descriptors and Directories

| | | |
|---|---|---|
| FS | block | block descriptor (BD) |
| | component | component descriptor (CMD)<br>component directory (CMDR) |
| | connection | connection descriptor (CND)<br>connection directory (CNDR) |
| | terminal | terminal descriptor (TD)<br>terminal directory (TDR) |
| | name | name descriptor (NMD)<br>name directory (NMDR) |
| | subscript | subscript descriptor (SD)<br>subscript directory (SDR) |
| PS | module | module descriptor (MD)<br>module directory (MDR) |
| | net | net descriptor (ND)<br>net directory (NDR) |
| | root | root descriptor (RD)<br>root directory (RDR) |
| | pin | pin descriptor (PD)<br>pin directory (PDR) |
| | through hole | through hole descriptor (THD)<br>through hole directory (THDR) |
| | coordinate | x-y coordinate descriptor (XD)<br>x-y coordinate directory (XDR) |
| | page | page directory (PGDR) |
| | assignment | assignment descriptor 1 (AD 1)<br>assignment directory 1 (ADR 1)<br>assignment descriptor 2 (AD 2)<br>assignment directory 2 (ADR 2) |

**A. Descriptors**

(i) block descriptor (BD)

Data for the FS are grouped together by blocks. A block descriptor points such a group of data, as given in Table 5. 2.

Table 5. 2 Block descriptor

| | |
|---|---|
| #1 | pointer to the CMDR of the relevant block. |
| #2 | pointer to the CNDR of the relevant block. |
| #3 | pointer to the NMDR of the relevant block. |
| #4 | pointer to the SDR of the relevant block. |

(ii) component descripter (CMD)

Each component has a component descriptor which contains data concerning its terminals, the block which is identical to ti, and the module to which it is assigned. See Table 5. 3.

Table 5. 3 Component descriptor

| | |
|---|---|
| #1 | pointer to the initial TD in the series of TD's for the relevant component. |
| #2 | pointer to the initial NMD in the linked list of terminal identifiers for the relevant component. |
| #3 | pointer to the BD of the block which is identical to the relavant component. |
| #4 | pointer to the MD of the module to which the relevant component is assigned. |
| #5 | pointer to the AD 1 giving the assignment of the relevant component. |

(iii) connection descriptor (CND)

Each connection has a connection descriptor which contains data concerning its terminals and the net corresponding to it. See Table 5. 4.

Table 5. 4 Connection descriptor

| | |
|---|---|
| #1 | pointer to a TD in the linked list of TD's specifying the relevant connction. |
| #2 | pointer to the ND of the net to which the relevant connection is assigned. |

(iv) terminal descriptor (TD)

Terminal descriptors are grouped together by components. They contain data as shown in Table 5. 5.

Table 5. 5 Terminal descriptor

| | |
|---|---|
| #1 | pointer to the CND which is incident to the relevant terminal. |
| #2 | pointer to the next TD in the linked list of TD's specifying the relevant connection. |

(v) module descriptor (MD)

Each module has a module descriptor which contains, as shown in Table 5. 6, data concerning its shape, its pins and the component(s) assigned to it.

Table 5. 6 Module descriptor

| | |
|---|---|
| #1 | the number of pins attached to the relevant module. |
| #2 | pointer to the initial PD in the linked list of PD's for the relevant module. |
| #3 | pointer to the initial XD in the series of XD's specifying the shape of the relevant module. |
| #4 | pointer to the AD 1 or AD 2 specifying the assignment of components to the relevant module. |

(vi) net descriptor (ND) and root descriptor (RD)

Each net has a net descriptor as shown in Table 5. 7. The root of the tree representing it is pointed by ND#1. If a net is not complete and is represented by

more than one rooted tree, root descriptors, as shown in Table 5. 8, are created to form a linked list of the roots.

Table 5. 7 Net descriptor

| | |
|---|---|
| #1 | pointer to the PD of the reet of the tree representing the relevant net, or pointer to the initial RD of the linked list of RD's for the rooted trees representing parts of the relevant nets. |
| #2 | pointer to the next subnet if the relevant net is large and partitioned. |
| #3 | pointer to the AD 1 specifying the assignment of connections to the relevant net. |

Table 5. 8 Root descriptor

| | |
|---|---|
| #1 | pointer to the PD of the relevant root. |
| #2 | pointer to the next RD in the linked list of RD's for the relevant net. |

(vii) pin descriptor (PD)

Each of the pins, through holes and branching points has a pin descriptor, as shown in Table 5. 9. In PD#1 and PD#2 are given the data concerning the net which is incident to it. PD#2 gives the link to the adjacent pin descriptor specified by the rooted tree which represents the net. PD#4 gives the x-y coordinates of the pin on the board. Pin descriptors are grouped together by modules. For each page of the board a linked list of pin descriptors is constructed using PD#6.

Table 5. 9 Pin descriptor

| | |
|---|---|
| #1 | ponter to the ND of the net which is incident to the relevant pin, through hole or branching point. |
| #2 | pointer to the PD of the adjacent pin, through hole or branching point in the rooted tree representing the relevant net. |
| #3 | pointer to the initial XD in the linked list of XD's specifying the routing of the relevant net. |
| #4 | the x-y coordinate of the relevant pin, through hole, or branching point on the board. |
| #5 | the layer number or pointer to the THD in the case of a through hole. |
| #6 | pointer to the next PD in the linked list of PD's for the pins, through holes and branching points contained in the same page. |

(viii) through hole descriptor (THD)

For a through hole, a through hole descriptor is attached to the pin descriptor of the through hole. The code of this descriptor is given at PD#5. Table 5. 10 shows a through hole descriptor.

Table 5. 10 Through hole descriptor

| | |
|---|---|
| #1 | the layer numbers of the layers which the relevant through hole connects. |
| #2 | pointer to the next PD in the linked list of PD's for the pins, through holes and branching points contained in same page. |

(ix) x-y coordinate descriptor (XD)

An x-y coordinate descriptor specifies a point on the board. This descriptor is given in Table 5. 11. A polygon representing a module is specified by a series of x-y coordinate descriptors. For a pin, through hole, branching point or turning point, an x-y coordinate descriptor is also generated to indicate its position on the board.

Table 5. 11 x-y coordinate descridtor

| | |
|---|---|
| #1 | the relevant x-y coordinate on the board. |
| #2 | pointer to the next XD in the linked list of XD's. |

(x) name descriptor (NMD) and subscript descriptor (SD)

When being input, the identifiers of the basic elements in the FS are registered for later use. Name descriptors and subscript descriptors, as shown in Tables 5. 12 and 5. 13 respectively, are created for this purpose. Name descriptors of components and connections, respectively, are linked in alphabetical order using pointers in ND#4. A linked list of descriptors is formed for terminals of each component. It a name has a subscript, its descriptor is linked to the corresponding subscript descriptor by using NMD#3. If more than one subscript interval are attached to a name, subscript descriptors corresponding to the intervals are created, and they are linked by using SD#4.

Table 5. 12 Name descriptor

| | |
|---|---|
| #1 | the relevant registered name. |
| #2 | the code of the identifier specified by the relevant name. |
| #3 | pointer to the initial SD in the linked list of SD's attatched to the relevant name. |
| #4 | pointer to the next NMD in the linked list of identifiers. |

Table 5. 13 Subscript descriptor

| | |
|---|---|
| #1 | the lowest of the relevant subscripts. |
| #2 | the highest of the relevant subscripts. |
| #3 | the number to be added to NMD#2 to find the codes of the identifiers specified by the relevant name and subscripts. |
| #4 | pointer to the next SD in the linked list of SD's atteached to the relevant name. |

(xi) assignment descriptor 1 and 2 (AD1 and AD2)

As stated in Section 2 C, a component at a leaf level is assigned to a module or a part of a module. This assignment can be followed using assignment descriptors 1 and 2, as shown in Table 5. 14. An assignment descriptor 1 is provided for each component at a leaf level, and the level and all the codes of the qualifying blocks of the component are included. If only one component is assigned to a module, the code of AD1 of the component is written down in MD#4 of the module. If more than one component are assigned to a module, an AD2 is created to indicate the assignement and its code is given in MD#4.

Table 5. 14 Assignment descriptors 1 and 2

| |
|---|
| AD1 |
| #1 1 plus the level number of the level at which the relevant element is. |
| #2 pointers to the CMD's of the elements qualifying the relevant element. |
| AD2 |
| #1 pointers to the AD1's of the components which are assigned to the module having the relevant pins. |
| #2 the terminal numbers assigned to the relevant pins. |

**B. Directories**

Directories contain the numbers of descriptors and some other information concerning the entities. The initial locations of the linked lists of descriptors for components and connections are given in the component directory and connection directory respectively.

**C. Relation Between the Functional Structure and the Physical Structure**

The links connecting the elements in the FS and PS are illustrated in Figs. 5. 1 and 5. 2.

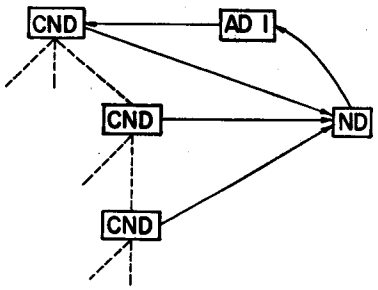


Fig. 5. 1. Links from connction descriptors to a net descriptor.

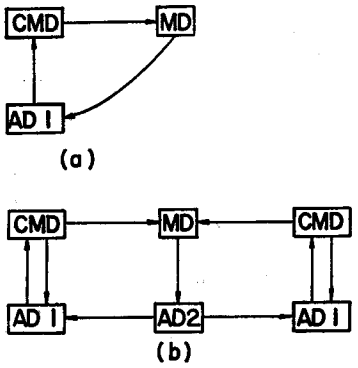


Fig. 5. 2. Links from a component descriptor(s) to a module descriptor. (a) one component to one module. (b) more than one component to one module.

## 6. Data Reference

In designing a circuit, a designer specifies the FS of the circuit. However, in the layout process, the display of the circuit is based on the PS. Thus he may want to know the element(s) in the PS related to an element in the FS or vice versa. He may also want to get information concerning the FS or PS itself. Therefore, in our system, the means are provided for referring to the structural data described in the previous section. The designer can specify the basic structural elements by the data reference formulae given below.

### A. Basic Data Reference Procedures

In our system, the basic data reference procedures are defined as given in Table 6. 1.

Table 6. 1 Basic data reference procedures

| | |
|---|---|
| BLK TO CMP: | gets all the components in the relevant block. |
| BLK TO CNN: | gets all the connctions in the relevant block. |
| BLK TO TRM: | gets all the terminals in the relevant block. |
| LGC TO BLK: | gets the block containing the relevant component, connction of terminal. |
| UPPER CMP: | finds the component inentical to the relevant block. |
| LOWER BLK: | finds the block identical to the relevant component. |
| CMP TO TRM: | gets all the terminals attached to the relevant component. |
| TRM TO CMP: | gets the component to which the relevant terminal is attached. |
| CNN TO TRM: | gets all the terminals conncted by the relevant connction. |
| TRM TO CNN: | gets the connction which is incident to the relevant terminal. |
| MDL TO PIN: | gets all the pins attached to the relevant module. |
| PIN TO MDL: | gets the module to which the relevant pin is attached. |
| NET TO PIN: | gets all the pins connected by the relevant net. |
| PIN TO NET: | gets the net which is incident to the relevant pin. |
| MDL IN LYR: | gets all the modules laid out on the specified layer. |
| PIN IN LYR: | gets all the pins laid out on the specified layer. |
| MDL TO CMP: | gets all the components assigned to the relevant module. |
| CMP TO MDL: | gets the module to which the relevant component is assigned. |
| NET TO CNN: | gets all the connections assigned to the relevant net. |
| CNN TO NET: | gets the net to which the relevant connction is assigned. |
| PIN TO TRM: | gets the terminal corresponding to the relevant pin. |
| TRM TO PIN: | gets the pin corresponding to the relevant terminal. |
| CMP TO PIN: | gets the pins attached to the part of the module to which the relevant component is assigned. |
| BLK TO MDL: | gets all the modules associated with the relevant block. |
| BLK TO PIN: | gets all the pins associated with the relevant block. |
| GIN TO MDL: | gets the module specified by the graphic cursor. |
| GIN TO NET: | gets the net specified by the graphic cursor. |
| GIN TO PIN: | gets the pin specified by the graphic cursor. |
| ID TO CODE: | gets the component, connection, or module specified by the given identifier. |
| SELECT ELM: | takes out the specified codes from a set of codes to form a new set. |

### B. Data Types

In regard to data reference, we assume that the data concerning a basic element in the FS or PS has a specific "data type." For example, we define BLOCK TYPE for a block and MODULE TYPE for a module. In addition to the data types defined for the basic structural elements, we define a data type GIN, which actually is the positional data specified by the graphic cursor. Basic data reference procedures generate data of one type from those of another.

### C. Data Reference Formula

In order to refer to certain data, the designer gives the system an instruction

Table 6. 2 Data reference formula

| |
|---|
| [reference formula] ::=<[expression]> |
| [expression] ::=[term]\|[expression] [dyadic operator] [term] |
| [term] ::=@\|@[number]\|#\|#[number]\|[identifier group]\|[qualifier] [identifier group]\| [terminal identifier]\|[qualifier] [terminal identifier]\|[qualifier] [terminal identifier]\| ([expression])\|[term]b[monadic operator] |
| [dyadic operator] ::=&\|!\|/ |
| [monadic operator] ::=BLOCK\|COMPONENT\|CONNECTION\|TERMINAL\|MODULE\| NET\|PIN\|UPPER\|LOWER\|LAYER([interval])\|SELECT([interval]) |

Note: The parts of the keywords not underlined can be omitted.

in the form of the data reference formula defined in Table 6. 2. Data reference procedures to be performed are specified by monadic operators. For example, if the designer inputs an instruction A BL MOD, BLOCK TYPE data of the block with identifier A are first obtained, and then MODULE TYPE data are referred by the basic data reference procedure BLK TO MDL. The data referred can be stored for later data reference. The stored data are identified by giving them an "@" mark plus a specific number. The data just referred to can be referred to again by the input of @0 or @.

The designer can specify an basic element of the PS displayed on the CRT using the graphic cursor. The reference formula for this specification begins with a "#" mark. For example, by # NET the NET TYPE data of the net specified by the cursor are referred. # MODULE can be abbreviated to # only. The layer on which the basic element is laid out is specified by the layer number following a #.

The designer can get, by specifying the dyadic operators "&," "!" and "/", the meet, the union and the difference, respectively, of sets of data already referred. The operand sets of data must be of the same type.

## 7. System Configuration

The total configuration of our system is shown in Fig. 7. 1. The host computer of the system is a FACOM M-200 installed at the Data Processing Center of Kyoto University. A TEKTRONIX color display unit 4027A is used as the display terminal. This unit is supported by the sub-routine package IGL supplied by the manufacturer. The host language of the system is PL/I. Details of the system are as follows.

(i) circuit data file

The circuit data file is a sequential file containing the data described in section 3.

(ii) CDL (circuit description language) interpreter

The first step of the layout design using our system is the input of the circuit structures. When the FS is inputted by using the CDL, a file of the structural data is generated by the CDL interpreter. The interpretation is performed block by block

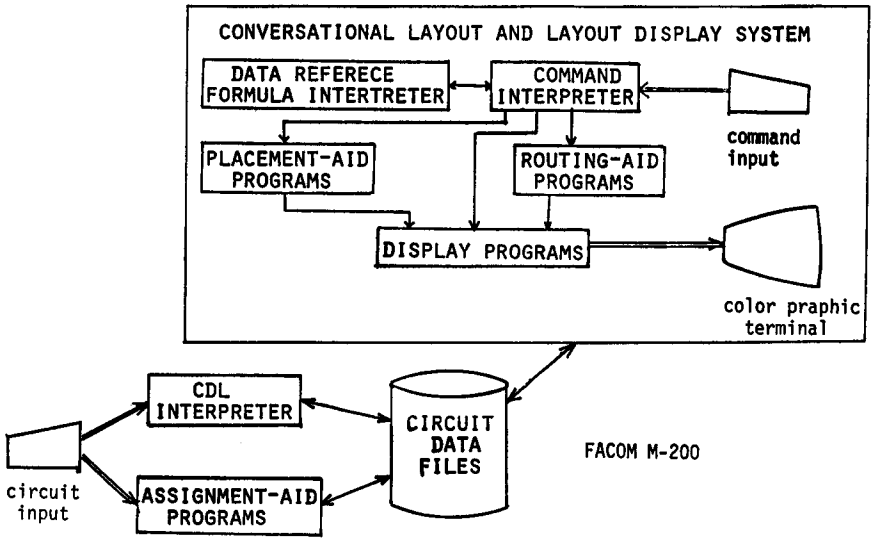


Fig. 7. 1. System configuration

as follows. First the block division is interpreted and a block descriptor is generated. Next the component division is interpreted, and for each component declaration a component descriptor is generated. The identifiers of components and terminals are registered by generating name descriptors. Then the interpretation of the connection division takes place, and connetion descriptors are generated. By the end of this step the generation of terminal descriptors is completed.

(iii) assignment-aid programs

The relation between the FS and PS is specified as the second step. This can be done with the help of the assignment-aid programs. The assignment of components to a module can be performed first by the input of the identifier of the module and then the identifiers of the components. Terminals of a component at a leaf level are numbered in the order as they are input. The assignment of terminals to pins are specified by these numbers and the result is written down in AD#2. Connections are assigned to a net by the system considering the hierarchy of the FS. CMD#4, #5 and CND#2 are filled in at this step.

(iv) conversational layout and layout display system

This sub-system is for the layout of the basic elements of the PS on the board conversationally. As the third step of the layout design, the designer determines the positions of modules, pins and external pins, and then the routing of nets by the input of layout and display commands from the graphic terminal. The laid out circuit is displayed on the CRT of the terminal. The commands for placement and routing are as follows.

1. PLACE: This command initiates the placement-aid programs and makes it possible for the designer to use placement sub-commands.
2. ROUT: This command initiates the routing-aid programs and makes it possible for the designer to use the routing sub-commands.
3. MOVE: This command moves a module from one position to another on the board. The nets connected to the module are removed automatically.
4. COPY: Copies of a module can be reproduced by using this command.

In general, the basic element to be placed, moved or routed is specified by a data reference formula given as the operand of a command. The designer can also ask the system to find an unrouted net with which he is to work on next. He specifies the routing pattern of a net using the graphic cursor. In this process of routing, branching points and through holes can be created or added.

The display commands which can be used for layout are as follows.

1. WINDOW: Using this command, the designer can specify the part of the board to be displayed. The size of the displayed area can also be given. If a basic element is specified as the operand of this command, the area to be displayed is automatically set up to include the element. It is possible to select a layer and display the elements on it.
2. COLOR: This command specifies the color of the elements on a layer.
3. DISPLAY: The basic elements specified as the operand of this command are displayed in the specified color.
4. IDENTIFIER: The identifier of the specified basic element is displayed.

The operands of the commands are data reference fomulae. Thus, by DISPLAY (A BL MOD) RED, for example, all the modules belonging to the block A are displayed in red.

## 8. Concluding Remarks

Using our system, the designer can have access to the laid out circuit with the knowledge of the FS, which makes the circuit layout much easier for him. For example, he need not trace electrical inter-connections on the circuit diagram and find out the wires which realize them on the board. Instead, he only has to specify some terminals connected to them, or has to specify a component if he wants all the inter-connections attached to it. The converse operation of finding the basic elements of the FS from the layout is also easy. The system takes full advantage of the color disply for easy recognition of specific elements, especially nets, laid out on a multilayer board.

Because of the rather complex data structures, the system requires a considerable amount of memory when the circuit size becomes large. Hence, the system needs to

be improved so that data can also be stored in external files. It is not possible to alter the FS once it is inputted. The addition of sub-programs which achieve changes in the FS, however, will not be difficult, since data concerning the FS are stored block by block using linked lists. No automatic placement or routing sub-routines are installed, and the system functions need to be expanded to handle large scale integrated circuits.

## References

1) Research Committee on the Design Technology of Electronic Equipments, "Computer Aided Design of Electronic Equipments (3)-The Recent Trends of the CAD Technology in Pysical Design," Data Processing, vol. 21, No. 1, pp. 50-61, Jan. 1980 (in Japanese).
2) Van Cleemput, W. M., "A Hierarchical Language for the Structural Description of Digital Systems," 14th D.A. Conference, pp. 377-385, 1977.
3) Preas, B. T, and Gwyn, C. W., "Methods for Hierarchical Automatic Layout of Custom Circuits Masks," Proc. of 15th D. A. Conference, pp. 206-212, 1978.
4) Sato, K., Nagai, T., Shimoyama, H. and Yahara, T., "MIRAGE-A Simple-Model Routing Program for the Hierarchical Layout Design of IC Masks," Proc. 16th D. A. Conference, pp. 297-303, 1979.
5) Chiba, T., Okuda, N., Kambe, T., Nishioka, I., Inufushi, T. and Kumura, S., "SHARPS: A Hierarchical Layout System for VLSI, Proc. 18th D. A. Conference, pp. 820-826, 1981.