ICOT Technical Memorandum: TM-0109

TM-0109

WIREX: VLSI Wiring Design Expert System

by
Hajimu Mori, Keiko Mitsumoto,
Tomoyuki Fujita, Satoshi Goto
(NEC Corporation), and
Hiroyuki Wakata
(NEC Information Service, Ltd.)

May, 1985

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

To be presented at VLSI 85, Tokyo

WIREX : VLSI Wiring Design Expert System

Hajimu Mori, Keiko Mitsumoto, Tomoyuki Fujita,
Hiroyuki Wakata* and Satoshi Goto

C & C Systems Research Laboratories
NEC Corporation
Kawasaki, JAPAN

*NEC Information Service, LTD.
Tokyo, JAPAN

This paper describes a new interactive routing system for VLSI layout design based on AI techniques. It adopts a knowledge-based method and relieves a designer from the burden of complicated design operations. This system incorporates the designer's knowledge as a set of rules represented in the Prolog language. To develop the knowledge-based routing system which can solve a large scale real problem, we have adopted a hybrid approach of combining the Prolog and FORTRAN languages. The system has been applied to design custom VLSIs containing several thousand gates and has shown quite promising results.

INTRODUCTIONS

The number of LSIs to be designed and the size of the LSIs are growing rapidly owing to recent advances in microelectronics technology. An 8-bit microprocessor design required 20 man-years, a 16-bit microprocessor design did 50 man-years, and for a 32-bit microprocessor design, more than 100 man-years is necessary. On the other hand, the number of LSI designers cannot increase so rapidly, especially expert designers, due to the extended training period which is necessary. LSI Design Expert Systems are expected to be able to accomplish innovation in this area. There are many research activities on artificial intelligence applications in LSI CAD. The system presented here is an expert system for solving a routing problem in VLSI layout design.

A routing problem in VLSI layout design is determining an appropriate connecting path. The path must meet all physical constraints for each given net. From a computational point of view, a routing problem is considered to be a hard combinatorial problem. Further, it obviously is a large-scale problem. The problem involves treating thousands of terminals of signal nets. Routing is the most time consuming phase in the entire LSI design process. It is far too difficult to complete the whole design by only using automatic programs with deterministic algorithms. The total design process consists of both automatic and manual design. In most practical cases, a large portion of the design time is spent on manual design, such as drawing, error checking and correction after automatic design.

In recent years, highly interactive CAD systems have been developed to cope with this problem [1,2,3] and the layout design time has been reduced a great deal. However, its successful operation is fully dependent on the designer's ability. The designer carries out the design by employing his own specific knowledge of the layout. The expert designer must examine the wiring patterns carefully on the graphic display to modify or create wiring patterns for complete net connection. This is a time consuming and error prone procedure.

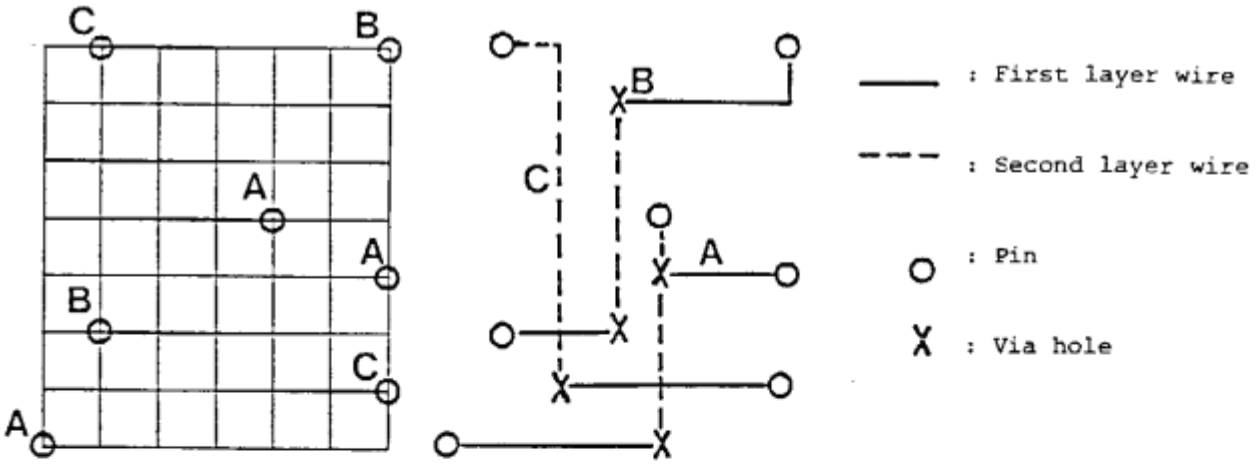
In the design process, an essential difference between human beings and the

computer is an intuitive ability for reaching an appropriate goal. The ability is considered to be based on a mechanism for applying "rule of thumb" reasoning. Recently, several new CAD systems, based on artificial intelligence techniques, have been reported [4,5,6,7,8]. However, it seems that there remain some difficulties in solving the large-scale real problems. The system proposed here is focused on applying the designer's specific knowledge to the design process for practical routing design problems. The designer's knowledge specifies a certain operation according to a specific situation, which is represented in the form of the design objects, their properties, and logical relations among them. In this system, the knowledge is expressed in a set of rules and written in the Prolog language [9] and stored in the knowledge database. The Prolog interpreter interprets the rules and makes inferences based on the knowledge to perform layout design task.

In order to make a computer an expert designer's assistant, this paper presents a new interactive routing system which applies the designer's specific knowledge to a practical design problem.

AI APPROACH TO VLSI ROUTING

A VLSI routing problem is to find a connecting path meeting all physical and electrical constrains for each given signal net. The connecting path is made by metal wires of two different layers and via holes for interlayer connections. A signal net is a connection requirement to be electrically equivalent for a set of pins. To simplify layout design, grid lines are introduced in the chip vertically and horizontally. Wiring patterns connecting pins can be routed only on these grid lines. Fig. 1(a) shows the connection requirement for the nets A, B, C, and one of the solutions is shown in Fig. 1(b).



(a) Connection requirement (b) Wiring layout

Fig.1 Routing problem

From the point of view of computational complexity, the routing problem is considered to be a hard combinatorial problem, in the sense that the computation time required to obtain the real optimum solution increases in exponential order, when the problem size, i.e., number of signal nets, increases. Consequently, the routing algorithms have been based on heuristic rationales. Although those heuristic algorithms have been improved year by year, they have not succeeded in producing a complete design in practical application.

The final routing goal is to achieve complete net connectivity in as short a design time as possible. To cope with this problem, interactive editing systems have been practically used in the wiring layout design (1,2,3). When 100% routing is not achieved by automatic programs, the incomplete routing result is

transmitted to the graphic display station for further completion. The designer has to create or modify wiring patterns on a CRT in order to achieve 100% routing. This is a time consuming and error prone procedure.

In the above interactive design process, the designer effectively employs knowledge acquired through experience or intuition and solves the problem. Therefore, if a part of the interactive design operations is replaced by automatic computer programs, the design time is expected to be extremely reduced.

Until now, artificial intelligence techniques have been actively developed and CAD engineers hope to use them for their own purpose. In this routing problem, the following key problems have to be clarified.

(1) Knowledge acquisition : What operations will the designer perform to solve a problem in a specific situation ?

(2) Knowledge representation : How should the designer's operation be represented in a knowledge-based form by a programming language ?

(3) Knowledge utilization : How should the knowledge database be utilized or exploited to solve a specific problem ?

We have analyzed the designer's operations for many cases and formalized them as a set of rules. We adopted Prolog as it is a language suitable for knowledge representation and utilization. However, there are two problems in current Prolog. Prolog requires a huge amount of memory space. The other problem is that Prolog runs to slow on the conventional computer to be practically used in the field such as LSI CAD. So, we cannot build a practical CAD system using only Prolog language on a conventional computer. Therefore, we have developed a new Prolog interpreter. This Prolog interpreter has a function which enables linkage with FORTRAN programs. This additional function can compensate for the conventional Prolog weak points and makes it possible to take advantage of existing FORTRAN programs. Most of the existing CAD systems are written in FORTRAN language. FORTRAN programs run fast and use memory space effectively.

SYSTEM CONFIGURATION

The Figure 2 shows a system configuration. The system includes two kinds of databases, knowledge database and CAD database. The knowledge database stores a certain amount of design knowledge about specific routing situations : properties of design objects, relations among the objects, and rules for guiding problem solving. The CAD database, which is a conventional database, contains layout design data, i.e., physical information about the design objectives. These two databases are accessed by the inference system and the CAD system respectively.

At present, the inference system is the Prolog interpreter. The Prolog language has an intrinsic inference mechanism, and the designer's knowledge is expressed directly in Prolog language. In this language, the rules can be considered as a set of knowledge and they are represented as a sequence of predicates. In order to have a linkage to the existing CAD system, our Prolog interpreter allows predicates that correspond to procedures in the CAD system. The procedure itself is written in FORTRAN statements and manipulates physical data in the CAD database.

Rules in Prolog programs have a hierarchical structure, and rules at the top level of the hierarchy can be considered as commands to the system. As the designer inputs a rule name, the interpreter interprets the program and produces a certain procedural sequence. This sequence represents the designer's operations on the display, and its execution can simulate the design process. These mechanisms enable the designer to solve a practical routing problem within a reasonable

amount of computer time and memory.

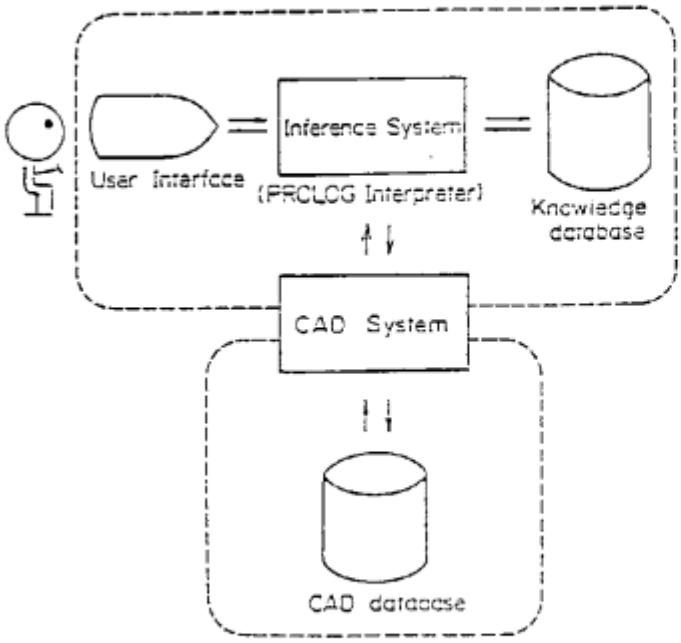


Fig.2 System configuration

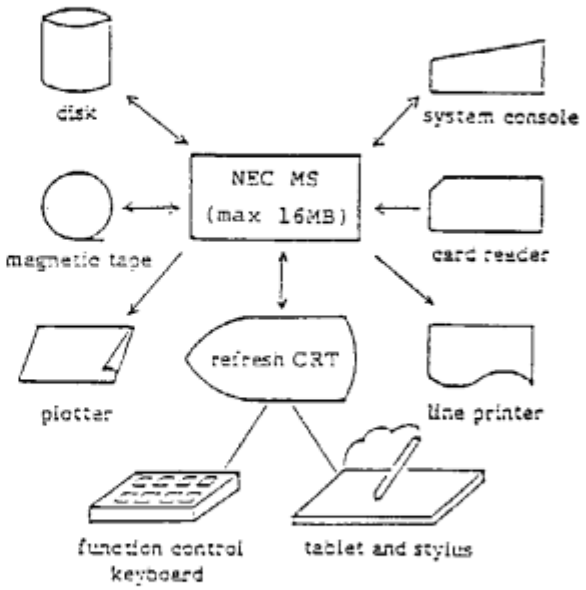


Fig.3 Hardware configuration

The system hardware configuration is shown in Fig. 3 and its picture in Fig. 4. The system operates on a NEC/MS minicomputer, which has up to 16M byte of memory capacity and a 200M byte disk storage capacity, coupled with a 21-inch, beam-directed refresh, graphic display. The display terminal includes a graphics tablet and a function control keyboard with 2048 by 2048 point address space.

Fig.4 WIREX system

PROLOG INTERPRETER

Prolog is one of the languages suitable for representing knowledge and developing rule-based systems. Prolog is characterized by the following three points ; the first point is to have a intrinsic inference mechanism. The second point is to be able to manipulate structured data objects, such as lists and trees, by applying pattern matching. The third point is to access program and data in the same way and allow them to be mixed together.

In general, the mechanism of a rule-based system is considered to be a simple repetition of selecting and executing rules. The rule-based system must operate pattern matching processes whenever such a repetition would occur. Therefore, as the rules contained by the rule-based system in the knowledge database increase, the inference speed decreases. In Prolog programming, data are expressed by clauses with empty body (unit clauses). The rule-based system has to store programs and many rules in the knowledge database to solve problems which have a large amount of data. This makes the inference speed slower.

To cope with this problem, procedural knowledge should be written in a more adequate language such as FORTRAN. Much of the knowledge in the design process is procedural knowledge, such as how to manipulate physical data and which design primitives should be executed. We have developed a Prolog interpreter which has the additional function of linking with the FORTRAN language. Design data in the CAD database of an existing CAD system can be accessed by FORTRAN routines. This is an efficient method for taking advantage of accumulated programs in an existing CAD system. Predicates corresponding to FORTRAN routines can be defined in the Prolog programs. Those routines are procedures in the CAD system, and they can access data in the CAD database. These predicates are called external functional predicates. They are preceded by '$' marks. This additional function enables the system to realize high speed processing with less computer memory.

When an external fuctional predicate is called in the unification process, the Prolog interpreter calls a FORTRAN subroutine. After execution, the Prolog interpreter receives the result (Fig. 5).

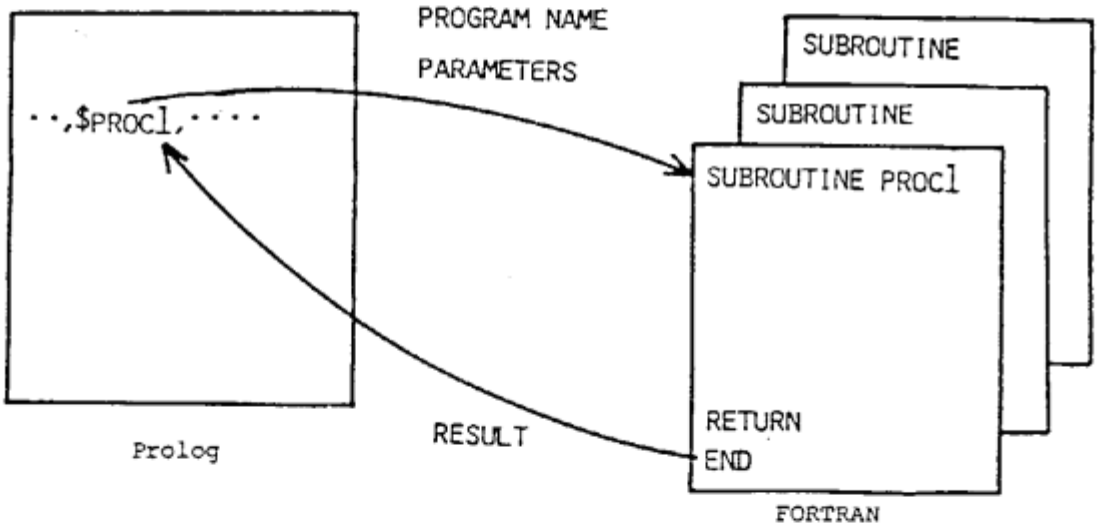


Fig.5 External functional predicate

An external functional predicate corresponds to a FORTRAN subroutine, which recovers data in the CAD database if the unification has failed in the following predicate. This recover function is introduced to keep the design result consistent with the CAD database, when the backtrack occurs in the Prolog clause.

KNOWLEDGE-BASED ROUTING

The designer's operations are carried out based on knowledge acquired through experience. In order to represent the designer's knowledge as a set of rules, the designer's operations must be analyzed.

After automatic routers have run, usually there are several uncompleted connections. When the designer trys to complete the net connectivity after using

automatic programs, he then understands the design situation. The designer devises a strategy to complete a net connection. Then he selects the appropriate commands.

The net connections are completed, pin-pair by pin-pair. The order is very important because unconnected pin-pairs influence each other. To determine which pin-pair should be tried first requires the know-how of the designer. He may select the pin-pair which is the farthest apart. He may connect net A before net B in the situation shown in Fig. 6, because if net B is connected first, wires of net B may block the connection of A. The system can have this knowledge in its knowledge-database.

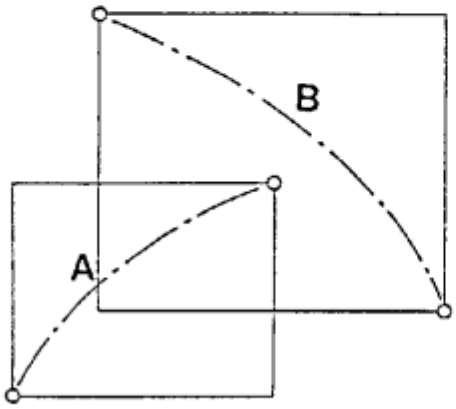


Fig.6 Pin-pair order

Unconnected connections are divided into two classes.
First class: Blocking wires exist closely around a terminal of the pin pair (Fig. 7).

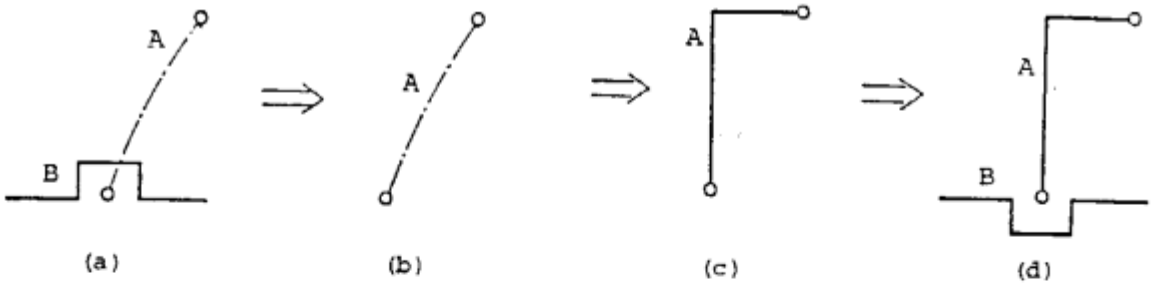


Fig.7 Blocking around a pin

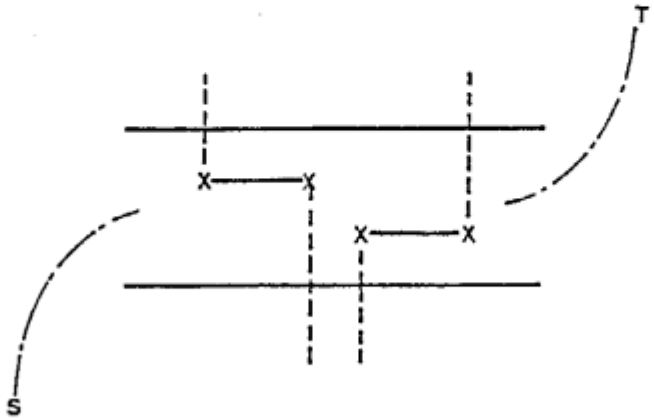


Fig.8 Capacity bottle-neck

Second class: There are capacity bottle necks (Fig. 8). We cannot connect a start point S and a target point T, because there is a very congested area along the route connecting the two terminals.

Figure 7 shows a simple example of the first class. An unconnected net is indicated by a dash-dotted line. Wire segments are indicated by solid lines. The unconnected net is blocked by other net wiring patterns on the neighboring grid line. In this situation, the designer can easily complete the net connection on the CRT display by the following operations.

(STEP1) Find net B, which blocks the connection of net A.
(STEP2) Alter the route of net B to allow the connection of A.
(STEP3) Connect net A.

These operations can be simulated by the following steps.

(STEP1) Find net B, which blocks the connection of net A.
(STEP2) Delete the net B wires.
(STEP3) Connect net A.
(STEP4) Connect net B.

These steps are considered to be the designer's knowledge to solve this specific problem. This knowledge is described as rules in Prolog. The following rule is one of them.

```
RULE30(*NET_A) :-
   BLOCKING(*NET_A,*NET_B),
   $DELETE(*NET_B),
   $CONNECT(*NET_A),
   $CONNECT(*NET_B).
```

Variables are represented by characters preceded by asterisks, and the external functional predicates are preceded by '$' marks. The predicate "BLOCKING" is also represented in detail as follows.

```
BLOCKING(*NET_A,*NET_B) :-
   GETPIN(*NET_A,*LAY,*XS,*YS),
   EQ(*LAY,1),
   DIRECTION(*YS,*DIR),
   $GRIDSEARCH(*DIR,*LAY,*XS,*YS,*NET_B).
```

The GETPIN predicate obtains coordinates (*XS,*YS) and a layer number *LAY of *NET_A. The EQ predicate checks whether the layer number is 1 or not. *DIR is a direction, which has four values (up, down, left and right), and the value is set after executing a DIRECTION predicate by unification. And $GRIDSEARCH predicate denotes a procedure which finds a blocking net *NET_B by searching the *LAY layer area toward the *DIR direction around the (*XS,*YS) coordinates. If one of the blocking nets is found, the next predicate $DELETE is tried. Otherwise, another blocking net is searched for by executing the BLOCKING rule clause.

When the goal clause 'RULE30' is given to the system, each predicate of the rule clause RULE30 is executed sequentially step by step. The predicate $DELETE corresponds to the DELETE routine in the CAD system, and the predicate $CONNECT to CONNECT routine. If the connection of net B cannot be completed, the predicate fails and backtracking occurs. Then, the alternative of net B is tried. Note that when the connection of net B is tried, data in the CAD database is already modified. When backtracking occurs, the data in the CAD database is recovered by the interpreter. This data-recovery mechanism is one of the features of our interpreter. The rule can simulate the designer's operations exactly to achieve complete net connectivity, as shown in Fig. 7.

There are rules corresponding to different situations in the first class. The system also has rules to determine which rule should be applied, and proposes the most suitable rule to solve each specific situation.

For the second class, the case of a capacity bottle-neck, the system has rules to determine good rough routes for unconnected nets. This is done by evaluating the congestion along the rough route candidates as shown in Fig. 9. The least congested route is proposed. When the operator rejects it according to his global strategy, the second least-congested route is proposed. When the operator accepts it, the capacity bottle neck in that route must be resolved. There are several rules for solving a capacity bottle-neck problem.

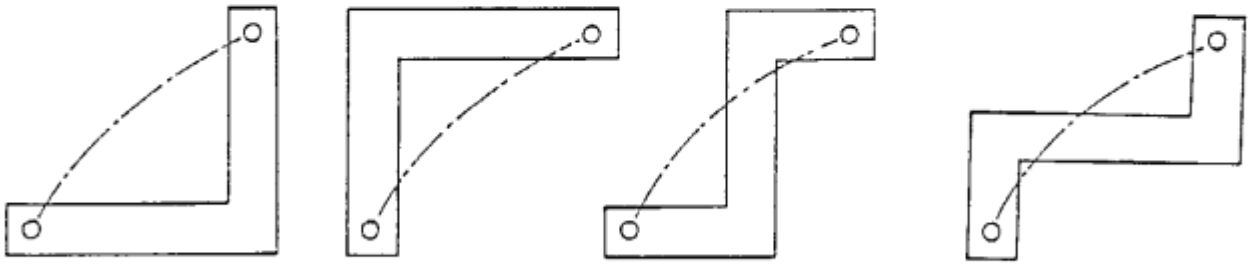
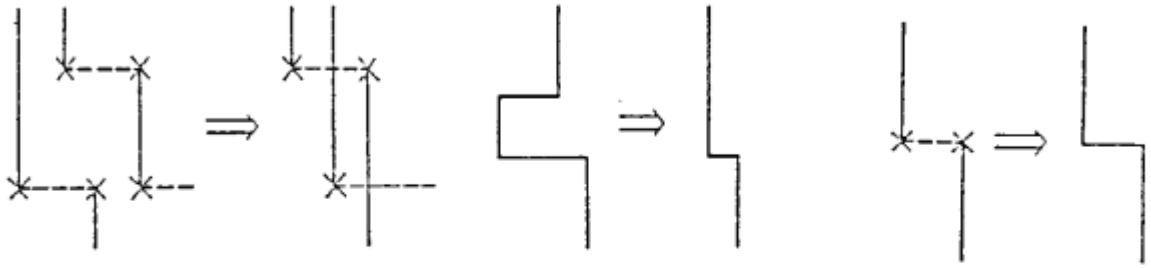


Fig. 9 Routing strategy

We have rules to determine which wire should be modified to connect pins S and T. Which rule should be applied is highly dependent on an individual situation. In one situation, a rule to select a short blocking wire segment is suitable, because the amount of modification is minimal. In another situation, a rule which selects a long one should be applied, because the possibility of connection completion is high.

There are also rules for pattern shape improvement. Wiring patterns are modified to have better shape by eliminating useless patterns like those shown in Fig. 10.



(a) altering wire routes (b) removing detour pattern (c) eliminating via holes

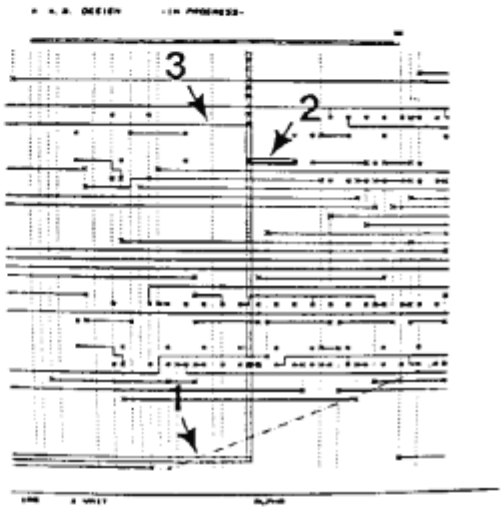
Fig. 10 Modifying wiring segments

In the knowledge-based routing system, rules or design strategies can vary adaptively according to design circumstances. For example, different structure or different size LSIs require different sets of rules or strategies. In this respect, knowledge-based systems have an advantage over conventional CAD systems.

Several examples are tried to evaluate the rules and inference mechanism. Figure 11 shows a routing result on the display, where two nets are left unconnected, as shown by dash-dotted lines. The system proposes a suitable rule to complete the connection of the net indicated by an arrow. When the designer accepts it, the

rule is applied to the net. Figure 12 shows the result, where one net is succeeded in connecting by executing the rule used in Fig. 11 routing situation. This LSI is a 2100-gate gate-array LSI. It has 2140 signal pins, which required 1448 conections. 12 connections could not be completed by the automatic router. With this system, the net connection was completed within half an hour by a nonexpert. No further complicated graphic operation is necessary to design this. The operator just pushes the yes-no buttons to indicate whether or not the strategy or rule provided by the system is acceptable. We eliminated trivial and tedious operations from the operator's work load. For the same problem, on the conventional interactive CAD system, it took more than one hour for an expert designer to complete the design. By accumulating more rules based on the designer's knowledge, more complex and difficult routing problems will be solved in a short time.

Fig. 11 Unconnected net



1. blocking wire
2. connecting path of an unconnected net
3. connecting path of the deleted net

Fig. 12 Execution result

Regarding the number of rules, we now have the situation shown in the table 1. There are 44 rules to solve a blocking problem near signal pins. There are 42 rules to solve a blocking problem involving a channel bottle neck. There are 15

Table 1 Number of rules

Rules to resolve blocking around pin	---	44
Rules to resolve catacity bottle-neck	---	42
Rules to modify existing pattern	---	15
Rules to select suitable rule	---	61
Rules to control the system	---	103
		265

rules to obtain improved pattern shapes. There are 61 rules to determine which rule is the most suitable in each situation. There are 103 rules for control, and most of them are rules for user interface such as graphic control rules.

These rules are stored in the knowledge database. The designer can also add new rules to the knowledge database and use them at the design stage, when he finds a good strategy to solve the particular situation.

CONCLUSION

This paper presents a new interactive routing system based on artificial intelligence techniques. We proposed, in a sense, a hybrid approach of combining the Prolog and FORTRAN languages. By doing this, we were able to construct a knowledge-based routing system which effectively takes advantage of an existing CAD system. The expert designer can store his own specific knowledge in the system and use it in the design process. The proposed knowledge-based system achieved an acceptable response time for the interactive operations and a reasonable result. By accumulating more rules, the system will become more intelligent and be able to solve more difficult problems.

An essential difference between human beings and a computer program is considered to be an intuitive ability for reaching an appropriate goal. In order to achieve fast and high-quality design, a CAD system should have a mechanism which incorporates the designer's knowledge. For VLSI design problems, many unsolved problems must be attacked which cannot be solved by using conventional techniques. We believe that the techniques presented here can be applied to numerous LSI CAD applications.

AKNOWLEDGEMENT

This research was supported by the Fifth Generation Computer Systems Project of the MITI in Japan. The authors would like to thank Dr. K. Furukawa of the Institute for New Generation Computer Technology (ICOT) for his valuable comments and suggestions. They are also greatly indebted to Drs. Y. Kato, T.Mikami and M.Naniwada for their helpful encouragement.

REFERENCE

[1] Skinner,F.D. ,Interactive Wiring System, Proc. of 17th Design Automation Conference (1980) 296-308.
[2] Mori,H., Fujita,T, Annaka,M., Goto,S. and Ohtsuki,T., Advanced Interactive Layout Design System for Printed Wiring Boards, in: Rabbat,G. (ed), Hardware and Software Concepts in VLSI (Van Nostrand Reinhold, 1983).
[3] Goto,S., Matsuda,T., Takamisawa,K., Fujita,T., Mizumura,H., Nakamura,H and Kitajima,K., LAMBDA : An Integrated Master-slice LSI CAD system, INTEGRATION 1 (1983) 53-69.
[4] Steel,G.L.,Jr. and Sussman,G.J., Constraints, Artificial Intelligence 14 (1980) 1-39.
[5] McDermott,J., Domain Knowledge and the Design Process, Proc. of the 18th Design Automation Conference (1981) 580-588.
[6] Brown,H. and Stefik,M, Palladio : An Expert Assistant for Integrated Circuit Design (Heuristic Programming Project Report HPP-82-5, Stanford, 1982).
[7] Steinberg,L.I. and Mitchell,T.M., A Knowledge Based Approach to VLSI CAD THE REDESIGN SYSTEM, Proc of 21st Design Automation Conference (1984) 412-418.
[8] Maruyama,F., Mano,T., Hayashi,K., Kakuda,T., Kawato,N. and Uehara,T., Prolog-Based Expert System for Logic Design, Proc. of the International Conference on Fifth Generation Computer Systems (1984) 563-571.
[9] Clocksin,W.F. and Mellish,C.S.: Programming in Prolog (Spring-Verlag, 1981).