A RULE-BASED COMPACTOR FOR VLSI/CAD MASK LAYOUT

Pei-Yung Hsiao, Chen Yung Syau, Wu-Shiung Feng
T. M. Parng and C. C. Hsu*

Institute of Electrical Engineering
*Institute of Information Engineering
National Taiwan University, Taipei, Taiwan, China

ABSTRACT

Compactor is a CAD tool used to pack rough mask diagrams to reduce the area size of the VLSI layouts. Often many iterations with human interventions are necessary to accomplish the manipulations of layout compaction. A rule-based system, in place of the conventional approaches of algorithms, is here proposed to explore the feasibility of using Expert System Technology to automate the compaction process by "reasoning" about the layout design using the sophisticated expert rules contained in its knowledge base.

I. INTRODUCTION

As we have known, layout compaction is one of the most challenging problems in VLSI layout systems. Since manual compaction is tedious, time-consuming, and error-prone, the automated compactor has turned out to play a more and more critical role in the CAD layout scheme. Therefore, in the past years, many algorithms have been developed to tackle the compaction problems[1]-[5]. For example, the three most popular algorithmic approaches include Compression-Ridge, Virtual-Grid, and Constraint-Graph based methodologies. However, the performances of these algorithms are still not comparable to those of human experts. From this point on, the authors had intended to induce a rule-based expert system (as an artificial expert) to solve the problem of layout compaction.

An expert system is a sophisticated computer program that manipulates knowledge to solve the problem efficiently and effectively by using symbolic logic and heurictics. Wherewith, this artificial expertise will bring us some advantages over human expertise. For instance, it is permanent, consistent, cheaper and modular. In this rule-based approach, rules containing domain knowledge can be added, deleted or modified without directly affecting the other rules. In general, a rule-based expert system consists of three major parts : A knowledge base composed of a set of production rules, a working memory indicating the current state of the system, and eventually an inference engine to control the system's activities.

In this paper, we have proposed a rule-based system, in place of the conventional approaches of algorithms, to explore the feasibility of using Expert System Technology to automate the compaction process by "reasoning" about the layout design using the sophisticated expert rules contained in its knowledge base. So far several experimental results have shown that our rule-based compactor is capable of producing more dense layouts which are competitive with human compacted results.

Even though there are some expert/knowledge based systems[8]-[11] proposed for problems of placement, routing and layout generation, a rule-based system published for mask layout compaction has not been found by us until now.

II. PRILIMITARY MODELING ON LAYOUT CONSTRAINTS

The set of constraints is dictated by the design rules, the user specified constraints (dubbed mixed constraints) and the implicit electrical circuits embedded in the layout. In this paper, we consider that constraints are generated between the x-coordinates of the left/right edges of the layout components. Remember that a component of the layout may be one object (i.e. rectangle, rhomboid or polygon) or one symbol (a set of intimate objects). Most of the constraints take the form

$$e_j - e_i > \lambda_{ij}, \qquad (1)$$

where the constraint, λ_{ij}, is a positive value, and e_i and e_j are the x-coordinates of the left/right edges of the same/different components. Besides, some other

constraints are brought up in the following forms

$$e_j - e_i = \lambda_{ij} \quad \text{or} \tag{2}$$
$$e_j - e_i < \lambda_{ij}, \tag{3}$$

here equation (3) is capable of being transfered into

$$-(e_j - e_i) > -\lambda_{ij}, \tag{4}$$

and equation (2) into

$$e_j - e_i \geq \lambda_{ij} \quad \text{and} \quad e_i - e_j \geq -\lambda_{ij}. \tag{5}$$

Lastly we then summarize all of the layout constraints from equation (1), (4) and (5) in an union form

$$\pm (e_j - e_i) \geq \pm \lambda_{ij}. \tag{6}$$

All of these constraints basically can be classified into three types. They are :

Type I : Width Constraints (λ_I).

Constraints set up from the left and right edges of the attentive component may consist of the min/max width constraints (> or <) and/or the frozen component constraints (=) of object/symbol. Such constraints, despite that they will come from design rule, user constraint or electrical requirement, are abstractly termed as a *width constraints* and are illustrated in Fig. 1.

Type II : Seperation Constraints (λ_{II}).

Constrains exsisting between the right edges of the left components and the left edges of the separated right components are called *separation constraints*. In short, "separation" means those two components which are never overlapped with each other. For a κ-visible compaction graph, the larger κ is, the much more **Type II** constraints must be concerned about and established as indicated in Fig.2.

Type III : Connection Constraints (λ_{III}).

From Fig.1, the majority of all of the constraints are grouped into *connection constraints* which denote all of the constraints formed between any two overlapped pair of components.

Mathematical Analysis

Definition 1 :

ψ is a geometrical operator such that if the x-coordinate of the left edge of rectangle R_j is not less than that of R_i, $i \neq j$, then the geometrical relationship between R_i and R_j is defined as $R_i \psi R_j$. In a formal manner, we define that

$$R_i \psi R_j : \; L_x(R_i) \leq L_x(R_j),\ i \neq j \text{ and } i,j = 1, 2, \ldots, N, \tag{7}$$

where $L_x(\cdot)$ is a function for evaluating x-coordinate of the left edge of the specified rectangle. ■

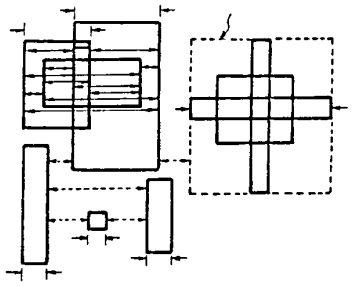


Fig. 1 Examples of type I (⟶|⟵), type II (<------->), and type III (⟷) constraints.

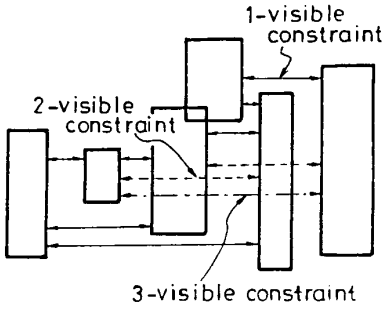


Fig. 2 Take each left edge as a shield edge. That implies the consept of κ-visible seperation constraint.

It is apparent that all of the λ_{II} and λ_{III} between any pair of the layout rectangles can be enumerated from $R_i \psi R_j$. All pairs of $R_i \psi R_j$ then can further be classified into three disjointed cases: ($R_i \psi_{III} R_j$), ($R_i \psi_{II} R_j$), and ($R_i \psi_{IV} R_j$). Rigoriously, these three cases are completely disjointed. Two of them are defined in the following definitions in a

formal way, and the last one will then be figured out without additional explanations. Whereas before doing that these three cases are illustrated in Fig. 3, 4, and 5 in order to help the understanding of their intuitively relational topology.

Definition 2 :

ψ_{III} is a geometrical operator which describes the connection relationship between two distinct rectangles. If $R_i \psi R_j$ and both of R_i and R_j are intersected each other, then we define the relationship between them as $R_i \psi_{III} R_j$. In summary we have

$$R_i \psi_{III} R_j : \quad R_i \psi R_j \ \wedge \ R_i \cap R_j \neq \{\}. \qquad (8)$$

■

lemma 1 :

For every pair of R_i and R_j, $i \neq j$, on the layout plane, if $R_i \psi_{III} R_j$, then there must exist at most three **type III** constraints, λ_{III}, between them.

Proof :

Recall that any type of constraints can be described by equation (6). It is not difficult to see that all of the constraints formed from design rules, user specifications or electrical requirements but settled up on the same couple edges of e_p and e_q, $p \neq q$, may be combined and reduced into an unique constraint in a form of equation (6). For any pair of rectangles, $R_i \psi_{III} R_j$, since there totally exist 4 distinct edges, the maximum amount of constraints possibly extracted from them are $C_2^4 = 6$.

Among these 6 constraints, 2 of them are belong to **type I** and the other 4 of them are dependent each other. From Fig.1 or Fig.4, it is easy to imagine that only 3 of the 4 remainded constraints are constraint independent. Hence we conclude the lemma.

■

Definition 3 :

ψ_{II} is a geometrical operator which presents one kind of the separation relationships between two distinct rectangles, R_i and R_j, such that

$$R_i \psi_{II} R_j : \quad R_i \psi R_j \ \wedge \ R_i \cap R_j = \{\} \ \wedge \ T_y(R_i) \geq B_y(R_j) \ \wedge$$

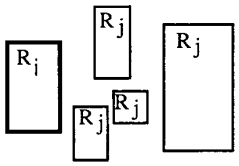


Fig. 3 $R_i \psi_{II} R_j$.

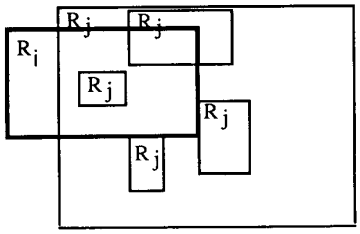


Fig. 4 $R_i \psi_{III} R_j$.

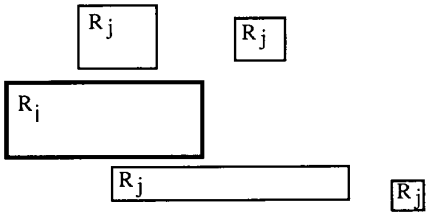


Fig. 5 $R_i \psi_{IV} R_j$.

$$B_y(R_i) \leq T_y(R_j) , \qquad (9)$$

where both $T_y(\cdot)$ and $B_y(\cdot)$ are functions for evaluating y-coordinates of the top and bottom edges of the specified rectangle, respectively.

■

lemma 2 :

For every pair of R_i and R_j, $i \neq j$, of the given layout, if $R_i \psi_{II} R_j$, then there must exist at most one **type II** constraint, λ_{II}, between them.

Proof :

From equation (6), all of constraints occurred in the same couple of edges, e_p and e_q, $p \neq q$, can be combined and reduced

into unique. Hence the λ_{II} exists between the right edge of R_i and the left edge of R_j to be one or none. Thus the lemma. ■

Property 1 :

According to lemma 2, the maximum possible amount of λ_{II} in a given layout is equal to N(N-1)/2.

Proof :

In the worst case, any pair of rectangles in the source layout may form a **type II** constraint, hence the maximum possible number of λ_{II} must be equal to $C_2^N = N(N-1)/2$. ■

Definition 4 :

For every $R_i\psi_{II}R_j$, if there is a rectangle R_k whose left edge is located between the right edge of R_i and the left edge of R_j, then the left edge of R_k is candicated as a *shield edge* between R_i and R_j. To give a formal definition of the shield edge, the following sufficient and neccessary conditions should be satisfied:

$$\begin{aligned} R_x(R_i) &\leq L_x(R_k) < L_x(R_j) \quad \wedge \\ T_y(R_k) &\geq MIN(\, B_y(R_i),\, B_y(R_j)\,) \quad \wedge \\ B_y(R_k) &\leq MAX(\, T_y(R_i),\, T_y(R_j)\,), \end{aligned} \tag{10}$$

where $R_x(\cdot)$ is a function for evaluating x-coordinate of the right edge of the specified rectangle. Hereby if we term the total amount of shield edges between R_i and R_j as $N_s(R_i, R_j)$, then an advanced geometrical operator, ψ_{II}^{κ}, is defined such that

$$\begin{aligned} R_i\psi_{II}^{\kappa}R_j : \quad & R_i\psi_{II}R_j \quad \wedge \\ & \kappa = N_s(R_i, R_j) + 1, \end{aligned} \tag{11}$$

where κ is the degree of visibility. ■

Property 2 :

From lemma 2, it is natural to define that the separation constraint with κ-visibility as *κ-visible separation constraint*, λ_{II}^{κ}, for $R_i\psi_{II}^{\kappa}R_j$ in corresponding to the λ_{II} for $R_i\psi_{II}R_j$. Therefore, we will obtain that $|\lambda_{II}| = \sum_{\kappa=1}^{N-1} |\lambda_{II}^{\kappa}|$, where $|\lambda_{II}|$ denotes the total amount of **Type II** constraints in the given layout.

Proof :

For any pair of $R_i\psi_{II}R_j$, since the total number of rectangles on the layout plane is N, the maximum possible $N_s(R_i, R_j)$ equals N-2. Thus according to equation (11), we conclude that

$$\begin{aligned} |\lambda_{II}| &= |\lambda_{II}^1| + |\lambda_{II}^2| + \cdots + |\lambda_{II}^{\kappa_{max}}| \\ &= |\lambda_{II}^1| + |\lambda_{II}^2| + \cdots + |\lambda_{II}^{(N-2)+1}| \\ &= \sum_{\kappa=1}^{\kappa_{max}} |\lambda_{II}^{\kappa}| = \sum_{\kappa=1}^{N-1} |\lambda_{II}^{\kappa}|. \end{aligned} \tag{12}$$

■

III. SYSTEM OVERVIEW

From the aforementioned three types of constraints, all the constraints stored in the *knowledge base* are either defined by the process technology or by the user. The source layout which is represented by a set of objects (or components) and the adjacency information on these objects should be collected in the *data base*. Both the knowledge base and the data base are consulted and ruled over by the *heuristic compaction strategy* (inference engine) that is an important and essential part of the rule-based compactor.

As referred to [6], a system overview of the rule-based system is constructed in Fig.6. This rule-based system precisely differs from an algorithmic compactor in

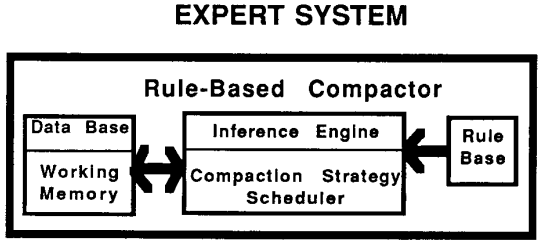


Fig.6 System overview of the rule-based compactor.

two respects : First, it is easily expandable and not restricted in a fixed algorithm. In other words, it is a fully heuristic scheme with involved extendability. Second, this rule-based compactor is not merely excuting two 1-D compactions one by one, but it also applies a number of expert rules to optimize the layout circuit performance and to fulfill many other miscellaneous requisitions : such as, minimizing the wire length, changing different layers and particularly processing the operation of via minimization.

In fact, we have observed that the key design issues and decision features comprised in the development of the expert system compactor are :

(A) System structure and control availability,

(B) Rule collection and knowledge representation, and

(C) Separation of program control and knowledge base.

For the reason that defining the system structure and the control scheme will cause some tradeoffs in deciding whether to program the LISP routines or to adopt the ready-made rules obtained from the OPS5 expert system language[7], we have succeeded in programming the control interpreter in LISP language and accomplishing the rule base by using OPS5's subprograms. This has proven to be an effective method, so that the coding design of OPS5 need not lead itself to trivially handle the initializations of program setup. In addition, the great benefit of the exchangable use of LISP programs, which normally perform the explicit calls to OPS5's solving subprograms, will gain more control availabilities than those being currently embedded in OPS5 language.

As described in the literature[7], the OPS5's working memory typically contains several hundred objects. Each object usually has 10 to 100 associated attribute-value pairs. The LHS (left-hand-side)[7] of a production consists of a sequence of patterns, which are taken for providing the partial descriptions of working memory elements. Whereas, the pattern matching and goal directed and data driven reasoning functions supplied by OPS5's language are used extensively throughout the system.

In considering the union of OPS5's rules and LISP's functions, the data flow architecture of the rule-based compactor is presented in Fig.7. In summary, the system scheme can be grouped into six ranks :

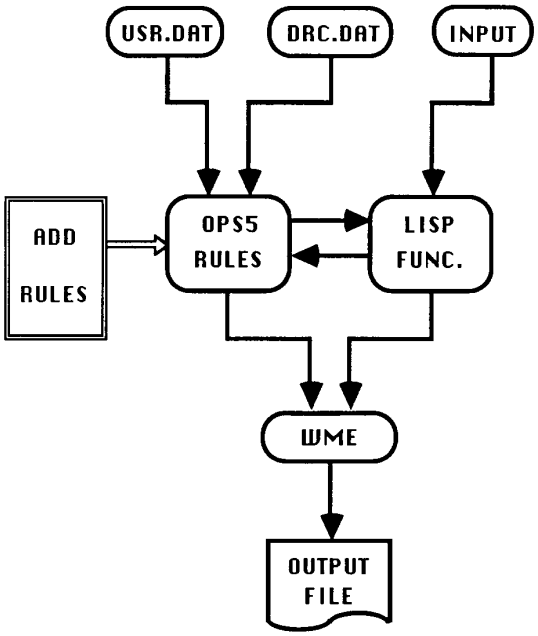


Fig.7 Data flow architecture for the expert system compactor

Rank 1: Prepare the technology design rules, the circuit electrical requirements, and the user specified constraints.

Rank 2: Create and define the vertex processes.

Rank 3: Keep away from any conflict among the constraints.

Rank 4: Generate pairs of constrained edges for design rule checking or constraint optimizing.

Rank 5: Reduce unnecessary slack to obtain a more packed layout.

Rank 6: Put the miscellaneous expert rules into practice.

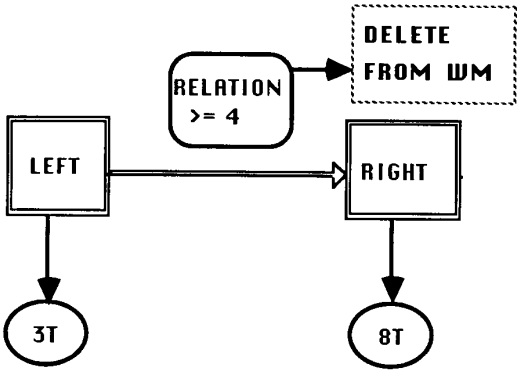
IV. KNOWLEDGE REPRESENTATION AND REASONING EXAMPLES OF THE EXPERT RULES

Since it is crucial for an expert rulebased system to select a proper knowledge representation, the programming of our system shares an essential characteristic: The knowledge tends to be hardwired in a sense that the exact patterns must be described in order for the program per se to perform the checking tasks. Also, for

this reason, the detailed knowledge of the program (perhaps with a little changing) has to add directly into the specific knowledge base.

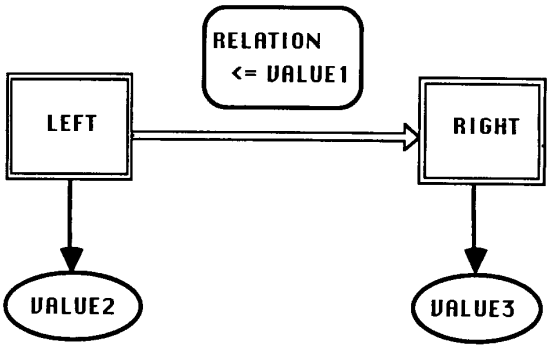
In this rule-based system, heuristics are encoded in rules and used to match the working memory. The nature of the separation of the rule base from the control part of the program allows an increment of expert rules without affecting the overall system design. As mentioned in the earlier section, the compaction process is grouped into ranks. The following examples of kernal rules shown in Fig.8, Fig.9 and Fig.10 precisely present the OPS5's rules used in our system. Those rules are able to match, create or delete the working memory elements (WME), while the LISP's functions will assert the WME into the specific working memory of the data base (i.e. input file or design rule file).

Here, despite the final result we will give in the later section, we have illustrated three instances in Fig.11, 12,and 13, for putting the matching conditions of misclaneous expert rules into practice.



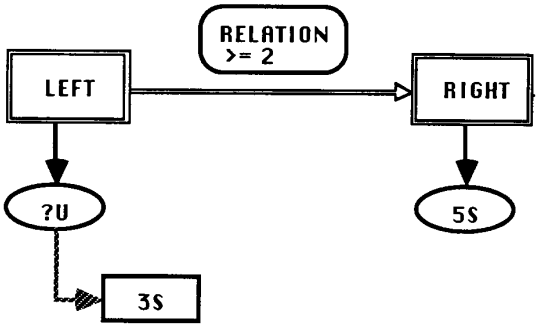

```
IF  THE LEFT NODE'S VALUE IS 3
      AND ITS TAG IS T
      AND THE RIGHT NODE'S VALUE IS 8
      AND ITS TAG IS T
      AND THE RELATION TYPE IS GREATED
      AND THE RELATION VALUE IS 4
THEN DELETE RELATION FROM WORKING
     MEMORY
```

Fig.8 Reasoning example for the kernel rule X.



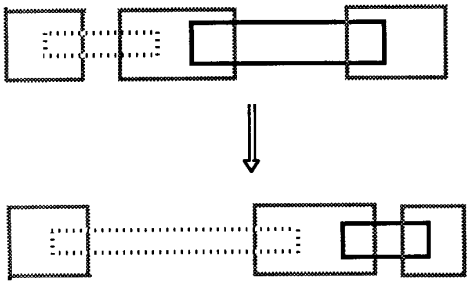

```
IF THE LEFT NODE'S VALUE IS VALUE2
    AND ITS TAG IS T
    AND THE RIGHT NODE'S VALUE IS VALUE3
    AND ITS TAG IS T
    AND THE RELATION TYPE IS LESSEQ
    AND THE RELATION VALUE IS VALUE1
THEN MODIFY THE RELATION TYPE TO GREATEQ
    AND MODIFY THE LEFT NODE'S VALUE TO
        VALUE3
    AND MODIFY THE RIGHT NODE'S VALUE TO
        VALUE2
    AND MODIFY THE RELATED CALUE TO
        -1*VALUE1
```

Fig. 9 Reasoning example for the kernel rule Y




```
IF THE LEFT NODE'S VALUE IS '?'
    AND ITS TAG IS U
    AND THE RIGHT NODE'S VALUE IS 5
    AND ITS TAG IS S
    AND THE RELATION TYPE IS GRATEQ
    AND THE RELATION VALUE IS 2
THEN MODIFY THE VALUE OF THE LEFT MODE
       TO 3
       AND MODIFY THE TAG OF THE LEFT
       MODE TO S
```

FIG.10Reasoning example for the kernel rule Z

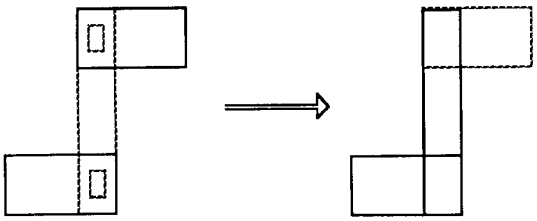


EXPERT RULE:

IF THE POLY WIRE IS TOO LONG TO PERFORM WELL
AND THERE IS A METAL WIRE WHICH CAN BE LENGTHENED

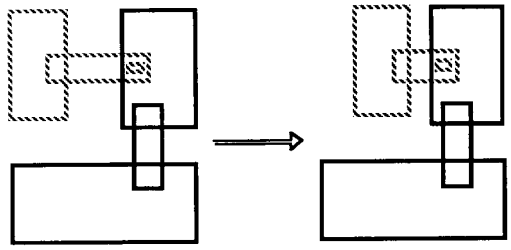
THEN LENGTHEN THE METAL WIRE
AND SHORTEN THE POLY WIRE

Fig.11 Example rule for optimizing the wire length.



EXPERT RULE:
IF THERE IS A VERTICAL SEGMENT OF A NET
CONNECING TWO VERTICAL SEGMENTS OF THE SAME NET
AND THE VERTICAL SEGMENT IS ON A DIFFERENT LAYER THAN THE TWO VERTICAL SEGMENTS
AND THE VERTICAL SEGMENT WHICH IS ON THE SAME LAYER AS THE TWO VERTICAL SEGMENT IS FREE AND CAN NOT BE USED BY ANY OTHER NET
THEN CHANGE THE LAYER OF THE VERTICAL SEGMENT OF THE NET TO THE SAME LAYER AS THE LAYER OF THE TWO VERTICAL WIRES

Fig. 12 Example rule for layer changing



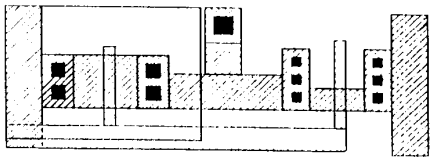
EXPERT RULE:
IF THE WIRE IS TOO LEFT AND TOO LONG
THEN SHORTEN IT

Fig. 13 Example rule for wire modification.

V. CONCLUSIONS

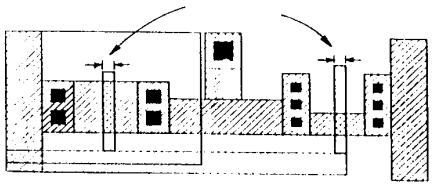
A novel rule-based layout compaction is proposed in this paper. It has been demonstrated that given a process independent mask layout, the design rules, user specified constraints and circuit electrical requirements can be checked over and rearranged into an optimized state. Fig.14 shows an example of the final result after the manipulations of compaction. Here, the first pair (pair (a)) of Fig.14 denotes an ordinary compaction in X-direction. Pair (b) presents that a supplementary expert rule of minimun width constraint is augmented for the circuit electrical requirements. Moreover, an additional λ_{II} (user-specified separation constraint) is put into consideration in the example of optimizing operation of pair (c).

This system currently consists of approximately 100 KBytes of source code and contains 120 rules. From the earlier discussion, this compactor is written in a combination of LISP and OPS5 languages. Our future researches include issues in introducing the C language to speed up the excution time and in extending the expert rules to enforce the system's intelligence. From our design experiences, it is not hard to see that it is more reasonable and acceptable to use the heuristic design technique based on the expert/knowledge based system to achieve a 2-D compaction (an NP-hard problem). Hence, in a word, it is the main interest of our future studies.

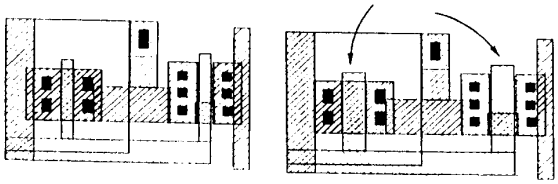


(a)

Minimum wire width constraint

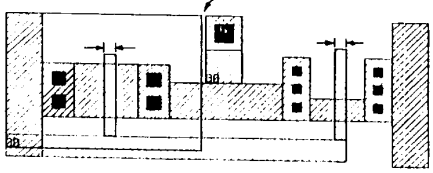


(b)

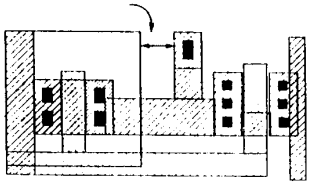


(a) (b)

User specified constraint



(c)



(c)

Fig. 14 Final result. Note that despite the design rules, this system also handles many miscellaneous constraints.
(a) Ordinary X-compaction.
(b) Compaction with external consideration of minimum width constraint.
(c) Optimization among miscellaneous expert rules.

REFERENCES

[1] Y.E. Chow,"A Subjective Review of Compaction," IEEE Proc. of 22th DAC, P. 396, 1985.

[2] P.Y. Hsiao and W.S. Feng, " An Edge-Oriented Compaction Scheme Based on Multiple Storage Quad Tree," accepted in IEEE Proc. of ISCAS, June 1988.

[3] P.Y. Hsiao and W.S Feng, " With M.S. Quad Tree on the Constraint Graph Compaction of The VLSI Large-Cell Layout-Editor," Proc. of VLSI TSA, p. 297, 1987.

[4] Jurgen Doenhardt and Thomas Lengauer, "Algorithmic Aspects of One Dimensional Layout Compaction," IEEE trans. on CAD, Vol. CAD-6, No. 5, p.863, 1987.

[5] J. F. Lee and D.T. Tang, " VLSI Layout Compaction with Grid and Mixed Constraints," IEEE trans. on CAD, Vol. CAD-6, No.5, p.903, 1987.

[6] D.A. Waterman, "A Guide to Expert Systems," Addson-Wesley Publishing Company, Inc., 1986.

[7] L. Brownston, R. Farrell, E. Kant, and N. Martin, "Programming Expert Systems in OPS5, " Addson-Wesley Publishing Company, Inc., 1985.

[8] T.J. Kowalski, " The VLSI Design Automation Assistant : A Knowledge Based Expert System," CMU, Report #CMUCAD-84-29, 1984.

[9] H. Cai, " A Rule-Based Global Routing Optimizer, " IEEE Proc. of ICCD, p.43, 1987.

[10] Y-L. S. Lin and D.D. Gajski, "LES: A Layout Expert System," IEEE Proc. of 24th CAD, p.672, 1987.

[11] T. Cesear, E. Iodice and C. Tsareff, " PAMS: An Expert System for Parameterized Module Synthesis," IEEE Proc. of 24th CAD, p.666, 1987.