Expert compactor: a knowledge-based application in VLSI layout compaction

P.-Y. Hsiao
C.-C. Tsai

Indexing terms: Computer-aided design, Very large scale integration

Abstract: A new application of artificial intelligence techniques in automatic compaction design for a VLSI mask layout is presented. To overcome the shortcomings of iterative search through a large problem space within a working memory, and therefore, to speed up the runtime of compaction, a set of rule-based region query operations and knowledge-based techniques for the plane sweep method are presented in this system. Experimental results have explored the possibility of using expert system technology to automate the compaction process by reasoning about the layout design, applying the sophisticated expert rules to its knowledge base.

1 Introduction

In years past, almost all of the published works on the applications of artificial intelligent (AI) technology to the problems of VLSI/CAD [18] were focused on verification [1], synthesis [2], placement [3], routing [4, 5], layout generation [6, 7, 8], and even an ASIC design [9]. Since manual compaction is tedious, time-consuming, and error-prone, many conventional algorithmic approaches for layout compaction [11, 12] have been presented in the past decades. For example, the three most popular algorithmic approaches include compression-ridge [22], virtual-grid [23], and constraint-graph [12–14, 10] based methodologies. Some of the one-dimensional algorithmic compactors are going to have mature products, however, the performances of these algorithms are never comparable to those of human experts. Based on this, the authors intended to develop a rule-based expert system as an artificial expert to solve the problem of layout compaction [26].

This artificial expertise will bring some advantages over human expertise. For instance, it is permanent, consistent, cheaper and modular. In this rule-based approach, rules can be added, deleted or modified without directly affecting the other sets of modularised rules. Moreover, the total amount of rule-based source code is less than that of the algorithmic compactor. So far, the experimental results have shown that our rule-based compactor is capable of producing dense layouts which are competitive with algorithmic compacted results.

Paper 7597E (C4, C6), first received 20th November 1989 and in revised form 4th June 1990

P.-Y. Hsiao is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050, Republic of China

C.-C. Tsai is with the Department of Electrical Engineering, National Taipei Institute of Technology, Taipei, Taiwan, Republic of China

2 System overview

The expert compactor is a modularised layout compactor targeted for double-metal N-well CMOS process technology. By replacing the set of process dependent design rules in the knowledge base, this compactor can be used in a process independent manner. The choice of X-compaction or Y-compaction as the first step, from which different final packed results may be obtained, is freely selected by the user.

Input description of the mask layout, which will be read into working memory by the functional operations in C language as shown in Fig. 1, is an object list with a layer number and absolute geometric locations. Before reading the input file, this system allows the user to choose optionally the cut and merge operation which is also written in C language. The cut and merge operation will rearrange some overlapped pairs of the source file.

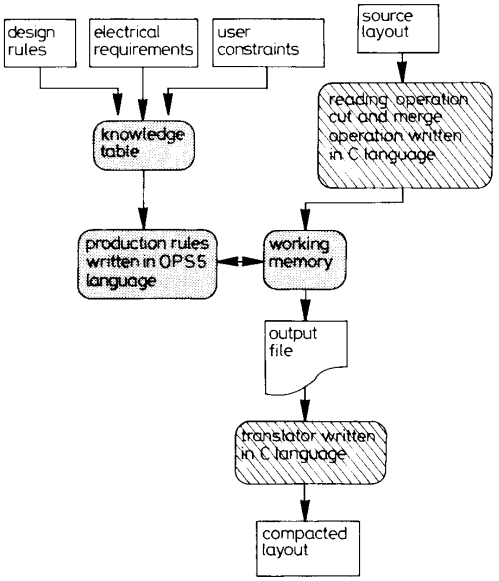


Fig. 1 *Data flow architecture for the expert compactor*

Fig. 2 presents four examples for the cut and merge operations. In addition, all of the design rules and user specified constraints are embedded as a knowledge table in the knowledge base through the rule representations in OPS5 [15–17]. The output file from the expert com-

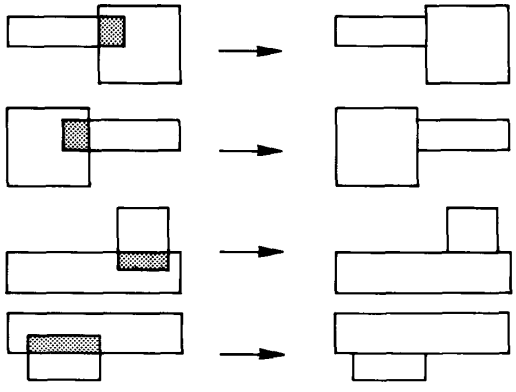


Fig. 2 *Examples for cut and merge operation*

pactor is translated and displayed as a physical layout through a new packed object-list to mask-layout translator, which is written in C language in References 14 and 24. Fig. 1 denotes the data flow architecture of the expert compactor. In summary, the system scheme can be grouped into six essential features:

(i) Preprocess the technology design rules, the circuit electrical requirements, and the user specified constraints.

(ii) Maintain a rule-based area localisation technique to speed up the runtime and a rule-based plane sweep method to encounter the layout objects effectively.

(iii) Generate pairs of constrained edges for design rule checking or constraint optimising.

(iv) Depress the redundancy effects of the conflict set from various constraints.

(v) Reduce unnecessary slack to obtain a more packed layout.

(vi) Take the miscellaneous expert rules into practice.

3 Knowledge-based space searching techniques

The study of data structures for area searching in 2D-space [19, 20] is a fascinating subject of practical interest in VLSI/CAD physical layout tools' design [13, 14, 21]. Instead of searching in a linear time order, most of the presented data structures perform the region query in a time order of $O(\log N)$, where N presents the total number of objects in 2D-space. A region query, frequently referred to as a 'pick' operation, will find all objects which intersect a specified region (window). Since the speed of such queries is crucial in many CAD applications, the efficiency of the adopted data structure becomes very important for the algorithmic approaches.

In addition to algorithmic approaches, the rule-based expert approach also needs some efficient heuristics to obtain the 'pick' operation at sheer speed.

Consider Fig. 3, there are twelve objects located on the layout plane. Five of them, shown in heavy solid lines are intersected by the specified window drawn in dashed lines. It is observed that there must be many possible relative geometric relations between window and each of the intersected objects. After collecting and classifying all of these relations, we get a summary of 205 possible relations. Without advanced analysing and extracting, it would be a painful experience to pull out the available expert rules from the large amount of possible relations.

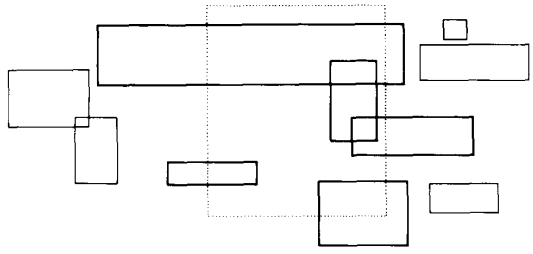


Fig. 3 *Example for region query*

Fortunately, five classes with sixteen disjointed cases from 205 possible relations have been encoded into the expert rules. The sampling expert rule for the first case of the first class is listed in Fig. 4*a–e*, where x_1, x_2, wx_1, and wx_2 indicate the x-co-ordinates of left and right edges of the layout object, R_r, as shown in solid lines, and the specified window, W_w, as shown in dashed lines.

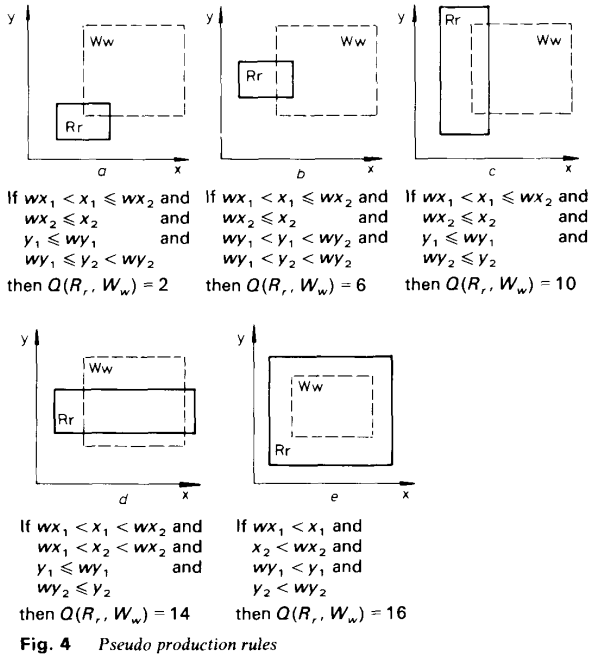


Fig. 4 *Pseudo production rules*

a Pseudo production rule for the first class of region query
b Pseudo production rule for the second class of region query
c Pseudo production rule for the third class of region query
d Pseudo production rule for the fourth class of region query
e Pseudo production rule for the fifth class of region query

By using rule-based region query operations to assist the implicit searching techniques of the pattern matching embedded in OPS5, the unavoidable slow runtime of knowledge-based techniques can be improved.

4 Knowledge-based plane sweep method

In the last decade, the plane sweep method has been a very popular technique for use in the algorithmic approach. Consider a vertical line sweeping from left to right in a two-dimensional layout plane. Since the X-co-ordinates of the layout plane are a set of continuous and infinite abscissa, the layout objects are fairly discrete

and finite, and it is necessary and much more desirable to transfer the continuous sweeping process to a discrete jumping operation. Although presorting the X- and Y-co-ordinates of all of the layout objects, respectively, could fit this requirement, it is not adaptable to our research work for two reasons. First, there is already a completely sorted tree structure embedded in the working memory elements of the knowledge engineering tool, OPS5 [15–17]. Sorting the data of the layout objects is a waste of time and is unsuitable for applying AI technology to our layout compaction scheme. Secondly, during the compaction process, the operations of region query are very often employed for recognising and coordinating the neighbouring relationships of constraints. Hence it is reasonable to imply the jumping operation to a rule-based approach by taking the sweeping line to be a particularly specified window of region query.

For a given layout, $\mathscr{L} = \{R_1, R_2, \ldots, R_N\}$, where R_i, $i = 1, 2, \ldots, N$, are N iso-oriented rectangles, Figs. 5, 6, 7, and 8 illustrate four distinguished cases for the current sweeping line, S_j, in the jth event to jump to the next sweeping event, $(j + 1)$th. Before implying case 1–4 of the jumping operation to the expert rules, the following definitions should be taken into account:

S_{jx}: X-co-ordinate of S_j.
$S_{(j+1)x}$: X-co-ordinate of S_{j+1}.
${}_jR_{act}, {}_jR'_{act}, {}_jR''_{act}, \ldots, {}_jR_{act,1}, {}_jR_{act,2}, \ldots$:
individual active rectangle for S_j.

Here we define the active rectangles as the rectangles intersected by the specified sweeping line, S_j.

m_j: the number of active rectangles corresponding to the current sweeping line, S_j.
A_j: set of the active rectangles for S_j, that means

$$A_j = \{{}_jR_{act,n} \mid n = 1, 2, \ldots, m_j\}.$$

For example, consider Figs. 5–8, we have

$$A_j|_{\text{Fig. 5}} = \{R_2, R_3, R_7, R_{10}\},$$

$$A_j|_{\text{Fig. 6}} = \{R_1, R_6, R_9\},$$

$$A_j|_{\text{Fig. 7}} = \{R_1, R_2, R_7\},$$

and

$$A_j|_{\text{Fig. 8}} = \{R_1, R_7, R_9\}.$$

$L_x(\cdot)$, $R_x(\cdot)$, $T_y(\cdot)$, and $B_y(\cdot)$: functions for evaluating the X-/Y-co-ordinates of the left, right, top, or bottom edges of the specified rectangle.

B_j: set of the nonactive rectangles neighbouring (in the right side to) A_j, such that $L_x(R) > S_{jx}$, in a formal manner, means

$$B_j = \{R \mid R \in \mathscr{L}, R \notin A_j, \quad \text{and} \quad L_x(R) > S_{jx}\}.$$

For example:

$$B_j|_{\text{Fig. 5}} = \{R_5, R_6, R_9\},$$

$$B_j|_{\text{Fig. 6}} = \{R_3, R_4, R_5, R_{10}, R_{11}\},$$

$$B_j|_{\text{Fig. 7}} = \phi,$$

and

$$B_j|_{\text{Fig. 8}} = \{R_2, R_3, R_8\}.$$

B'_j: a subset of B_j such that

$$B'_j = \{R \mid R \in B_j \quad \text{and} \quad L_x(R) < \text{MIN}\{R_x({}_jR_{act,n}) \mid n = 1, 2, \ldots\}\}.$$

For example:

$$B'_j|_{\text{Fig. 5}} = \phi,$$

$$B'_j|_{\text{Fig. 6}} = \phi,$$

$$B'_j|_{\text{Fig. 7}} = \phi,$$

and

$$B'_j|_{\text{Fig. 8}} = \{R_2, R_3\},$$

The expert rules shown below for plane sweep are written in pseudo OPS5 language.

```
(p evaluate-next-sweep-case1
;see Fig. 5.
   {∃ R_b ∈ A_j ⇒ R_x(R_b) > S_jx}
```

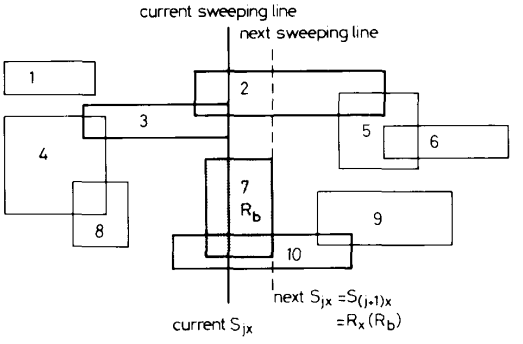


Fig. 5 *Example of case 1*

```
;choose the candidated active rectangles, one
;by one.
   {∀ _jR_act ∈ A_j, _jR_act ≠ R_b ⇒ R_x(R_b) ≤ R_x(_jR_act)}
;the accepted active rectangle, R_b, must have a
;smallest X-co-ordinate of the right edge.
   {∀ R ∈ B_j ⇒ R_x(R_b) ≤ L_x(R)}
;the X-co-ordinate of the left edges of all of the
;rectangles belong to B_j must be larger than or
;equal to the X-co-ordinate of the right edge of the
;accepted active rectangle.
   →
   (next S_jx = S_(j+1)x = R_x(R_b))
;the X-co-ordinate of the next sweeping line is set
;as the X-co-ordinate of the right edge of the acce-
;pted active rectangle, R_b.
)
(p evaluate-next-sweep-case2
;a disruption exists in the layout, see Fig. 6.
   {∀ _jR_act ∈ A_j ⇒ R_x(_jR_act) = S_jx}
```

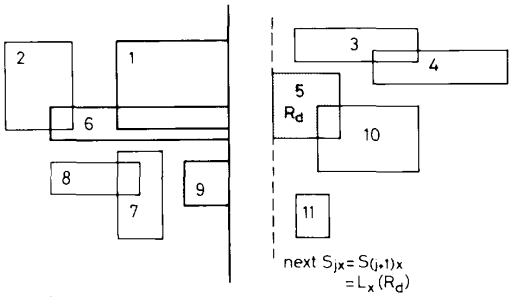


Fig. 6 *Example of case 2*

;$B'_j = \phi$ and all of the right edges of the current
;active rectangles are encountered by the current
;sweeping line.

$\{\exists\ R_d \in B_j$ and $\forall\ R'_d \in B_j \Rightarrow L_x(R_d) \leqslant L_x(R'_d)\}$

;choose $R_d \in B_j$ such that R_d has a smallest
;X-co-ordinate of left edge.

→

(next $S_{jx} = S_{(j+1)x} = L_x(R_d)$)

)

(*p* evaluate-next-sweep-case3
;end of sweeping, see Fig. 7.

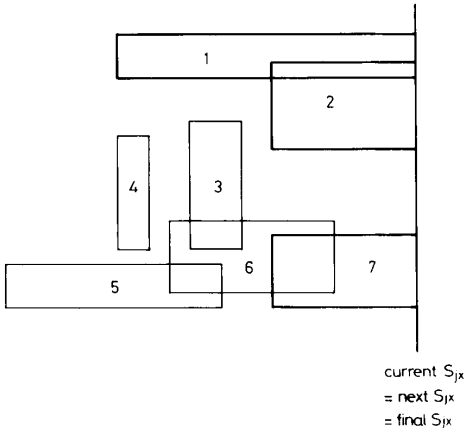


Fig. 7 *Example of case 3*

$\{\forall\ {}_jR_{act} \in A_j \Rightarrow R_x({}_jR_{act}) = S_{jx}\}$

$\{B_j = \phi\}$

→

(Stop sweeping)

)

(*p* evaluate-next-sweep-case4
;see Fig. 8.

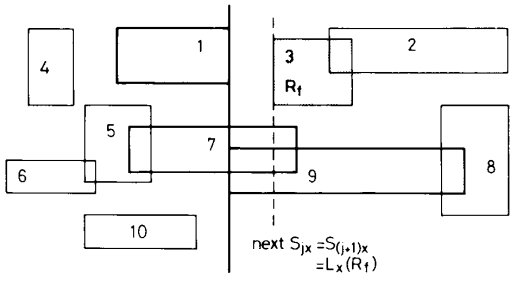


Fig. 8 *Example of case 4*

$\{\exists\ {}_jR_{act} \in A_j \Rightarrow R_x({}_jR_{act}) > S_{jx}\}$

$\{\forall\ {}_jR'_{act} \in A_j \Rightarrow R_x({}_jR_{act}) \leqslant R_x({}_jR'_{act})\}$

$\{\exists\ R_f \in B_j \Rightarrow L_x(R_f) < R_x({}_jR_{act})\}$

;choose the candidated $R_f \in B_j$s one by one, where
;the X-co-ordinate of the left edge of B_j must be
;smaller than any one of the current active
;rectangles.

$\{\forall\ R'_f \in B'_j \Rightarrow L_x(R_f) \leqslant L_x(R'_f)\}$

;the X-co-ordinate of the left edge of the accepted
;$R_f \in B_j$ must be smaller than or equal to any one
;of B_j.

→

(next $S_{jx} = S_{(j+1)x} = L_x(R_f)$)

)

5 Constraint formulation

Before giving an example study for applying the knowledge-based plane sweep technique to the knowledge domain of the compaction scheme in the sixth section, the prerequisite mathematical model of the layout constraints will be formulated in this Section.

The layout constraints are formed from the design rules, the user specified constraints and the implicit electrical requirements embedded in the layout. In this paper, we will focus our arguments on the horizontal constraints which are generated between the X-co-ordinates of the left and right edges of the rectangles. Recall that every rectilinear rectangle has been partitioned into rectangles before compaction. In a similar way, the vertical constraints used for Y-compaction can be established. Most of the constraints take the form

$$e_q - e_p \geqslant \lambda_{pq}, \tag{5}$$

where the constraint λ_{pq} is a positive value, and e_p and e_q are the X-co-ordinates of the left/right edges of the same/different rectangles. Besides, some other constraints are brought up in the following forms:

$$e_q - e_p = \lambda_{pq} \tag{6}$$

or

$$e_q - e_p \leqslant \lambda_{pq}. \tag{7}$$

Here eqn. 6 is capable of being transferred into

$$e_q - e_p \geqslant \lambda_{pq} \quad \text{and} \quad e_p - e_q \geqslant -\lambda_{pq}, \tag{8}$$

and eqn. 7 into

$$e_p - e_q \geqslant -\lambda_{pq}. \tag{9}$$

Accordingly, we then summarise all of the layout constraints from eqns. 5, 8 and 9 into a union form of

$$e_q - e_p \geqslant \lambda_{pq} \quad \text{and/or} \quad e_p - e_q \geqslant -\lambda_{pq}. \tag{10}$$

All of these constraints can be basically classified into three types. They are described in the following.

Type I: width constraints (λ_1).
Constraints set up from the left and right edges of the attentive object may consist of the min/max width constraints ($>$ or $<$) and/or the frozen constraints ($=$) of objects. Such constraints, despite the fact that they will come from design rules, user specified constraints or electrical requirements, are abstractly termed width constraints and are illustrated in Fig. 9.

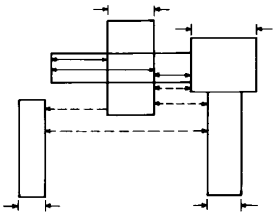


Fig. 9 *Sampling of type I (→\| \|←), II (← - - →) and III (←——→) constraints*

Type II: separation constraints (λ_{II}).
Constraints existing between the right edges of the left objects and the left edges of the separated right objects are called separation constraints. In short, 'separation' means those two objects are never overlapped with each other.

Type III: connection constraints (λ_{III}).
From Fig. 9, the majority of all of the constraints are grouped into connection constraints, which denote all of the constraints formed between any two intersected pairs of objects.

To fully exploit the relationship between any pair of intersected objects, the connection constraints are divided into four classes: ${}_1\lambda_{III}$, ${}_2\lambda_{III}$, ${}_3\lambda_{III}$ and ${}_4\lambda_{III}$, as shown in Fig. 6. Among these four classes of type (III) constraints, the first three of them are independent, and ${}_4\lambda_{III}$ is always absent in most real process technology. The following property gives us evidence to show the great consequence of ${}_1\lambda_{III}$, ${}_2\lambda_{III}$ and ${}_3\lambda_{III}$.

Property:
For every intersected pair of R_i and R_j, $i \neq j$, on the layout plane, there must exist at most three type (III) constraints between them.

Proof of Property:
Recall that any type of constraint can be described by eqn. 10. It is not difficult to see that all of the constraints formed from the design rules, user specified constraints 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 a unique constraint in the form of eqn. 10. For any pair of intersected objects, since totally there exist four distinct edges, the maximum number of constraints possibly extracted from them is $C_2^4 = 6$. Among these six constraints, two of them belong to type (I) and the others are dependent on each other. From Fig. 10, it is expected that only three of the four remaining constraints are constraint independent. Hence we conclude the property.

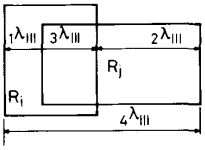


Fig. 10 *Type III constraints between R_i and R_j*

6 Domain knowledge and reasoning examples of expert compactor

As mentioned above, besides providing a random space searching, the expert region query techniques were successfully taken to build a rule-based plane sweep operation through which the layout objects were thoroughly traversed in a systematic manner. Between each jumping step of the sweeping line, some traversed layout objects will be encountered at their left or right edges. At that time, a great number of neighbouring constraints must be checked over by employing domain knowledge to obtain a dense layout. Wherein the neighbouring constraints are localised by region query techniques. In this Section, after describing the rule-based sweeping operation, efforts will be given to attempt to present an example of the width compaction case for knowledge representations and how to reason out their related expert rules.

Fig. 11 shows a case example of width compaction named case A. Consider the possible geometrical relationship between R_i and R_j, wherein the left edge of R_i is presently intersected by the sweeping line, so that R_i is named the master object. For each master object, R_i, there exists a number of slave objects, R_j, $j = 1, 2, 3, \ldots$, which intersect R_i in a geometrical manner just like the sampling illustrations in Fig. 12. Whenever the case A is met the LHS (left-hand side) of the rule of the start rule for case A shown below is matched and the actions of the RHS (right-hand side) will be activated.

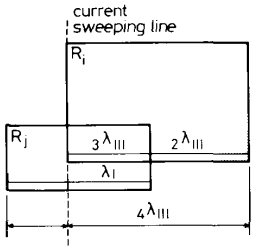


Fig. 11 *Typical example of case A for presenting techniques of knowledge elicitation*

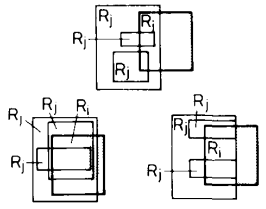


Fig. 12 *Sampling of case A*

(p start-rule-for-case$A$

{goal-width-R_i}

;control goal for compacting the width of R_i,
;$\lambda_I(R_i)$, namely, the slant line area in Fig. 11.

$\{L_x(R_i) < R_x(R_j) < R_x(R_i)\}$

$\{L_x(R_j) < L_x(R_i)\}$

$\{T_y(R_j) \geq B_y(R_i)\}$

$\{B_y(R_j) \leq T_y(R_i)\}$

;check whether there exist case A between R_i and
;R_j by pattern matching for expert region query
;techniques.

→

(Get ${}_2\lambda_{III}(R_j, R_i)$, ${}_3\lambda_{III}(R_j, R_i)$, ${}_4\lambda_{III}(R_j, R_i)$,
and $\lambda_I(R_j)$ from the constraint knowledge table.)

(Let $\Delta x_1 = [R_x(R_j) - L_x(R_j)] - \lambda_I$,

$\Delta x_2 = [R_x(R_i) - R_x(R_j)] - {}_2\lambda_{III}$,

$\Delta x_3 = [R_x(R_j) - L_x(R_i)] - {}_3\lambda_{III}$,

and

$\Delta x_4 = [R_x(R_i) - L_x(R_j)] - {}_4\lambda_{III}$.)

;assign possible compacted units to the delta
;x-variables concerned with λ_I, ${}_2\lambda_{III}$, ${}_3\lambda_{III}$, and
;${}_4\lambda_{III}$, respectively.

(Make goal-case$A$-R_j)

;trigger the control goal for case A in conjunction
;with one of the slave objects of R_j.
) ■

After activating the subgoal of goal case$A$ R_j, two most important expert rules shown in the following for responding to the possible candidate compaction distances will be triggered. Finally, by analysing all of the candidated compaction distances, an acceptable set of compaction distances will be extracted to perform physical compaction.

(p rule1-for-case$A$

{goal-case$A$-R_j}

{MIN (Δx_1, Δx_3) $\geq \Delta x$ or MIN (Δx_2, Δx_4) $\geq \Delta x$)}

;width of R_i can be completely compacted into mini-
;mum width.
→

(return-value-for-$R_x(R_i) = \Delta x$)

;candidated compaction distance for the right edge
;of R_i.

(return-value-for-$R_x(R_j)$ = MIN (Δx_1, Δx_3))

;candidated distance for the right edge of R_j.

(Remove goal-case$A$-R_j)

;end of case A for R_j.
) ■

(p rule2-for-case$A$

{goal-case$A$-R_j}

{$0 \leq$ MIN (Δx_1, Δx_3) $\leq \Delta x$}

{$0 \leq$ MIN (Δx_2, Δx_4) $\leq \Delta x$}

;width of R_i may only be partly compacted.
→

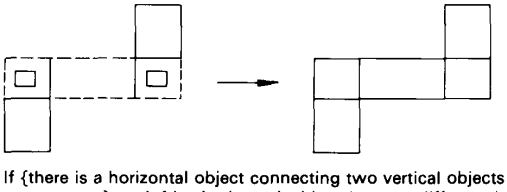
(return-value-for-$R_x(R_i)$ = MIN {[MIN (Δx_1, Δx_3)
+ MIN (Δx_2, Δx_4)], Δx})

(return-value-for-$R_x(R_j)$ = MIN (Δx_1, Δx_3))

(Remove goal-case$A$-R_j)

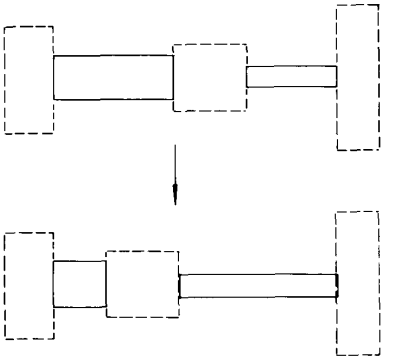
;end of case A for R_j.
) ■

Our system involves a lot of sets of expert rules for different cases which are extracted in a similar way but are limited to being enclosed one by one in this paper. Moreover, it is evident that an exhaustive traversal of all the possible geometric relationships among the layout objects will ensure a correctly compacted result. Despite the quick search and efficient jumping line techniques, the system will generally run slower. However, to obtain a denser layout depends on additional miscellaneous expert rules by which the performance of the expert compactor can be promoted to compete with a human expert. Here, in addition to presenting final results in the next Section, two examples for putting the matching conditions of extended miscellaneous expert rules into practice is displayed in Figs. 13 and 14.



If {there is a horizontal object connecting two vertical objects in the same net} and {the horizontal object is on a different layer in comparing with the two vertical objects} and {the horizontal object which is on the same layer as the two vertical objects is free and was not used by any other net}
then
(change the layer of the horizontal object into the same layer as the layer of the two vertical objects)
(remove the two contact objects)

Fig. 13 *Pseudo expert rule for layer rearrangement*



If {the right solid wire is too short to perform well} and {there is a different left solid wire which is allowed to be shortened}
then
(shorten the left solid wire)
(move the connected dash object)
(lengthen the right solid wire)

Fig. 14 *Pseudo expert rule for optimising the wire length*

7 Results and conclusions

This paper presents a rule-based expert compactor, in place of the conventional approach of algorithms, to explore the possibility of using expert system technology to automate the compaction process by reasoning out the layout design and applying the sophisticated expert rules to its knowledge base. The novel rule-based expert compactor has demonstrated that for a process independent mask layout, first, the set of design rules are examined by efficient region query operations and a knowledge-based area localisation of the plane sweep method, then the user specified constraints and the circuit electrical requirements are checked over to rearrange the source layout into a denser size.

Fig. 15 presents an example of the experimental result. The source layout shown in Fig. 15*a* contains more than 24 transistors. The compacted result gives an area reduction of 67.5% in a runtime of about 2,000 seconds and is shown in Fig. 15*b*.

To compare with performance of other compaction techniques, Fig. 16 presents another layout example selected from the Boyer's benchmark [25]. The source layout, the compacted result from the expert compactor, and the compacted result from the algorithmic compactor presented in References 14 and 24 are illustrated in Fig. 16*a*, *b* and *c*, respectively. The performance shown in Fig. 16*d* will support our claim that this expert compactor is competitive with the algorithmic compactor.

The system consists of approximately 3,000 lines of source code and contains about 200 rules. To speed up the execution time, the newest version of this system is currently settled on a SUN 3 workstation. In this rule-based approach, the expert rules can be easily added,

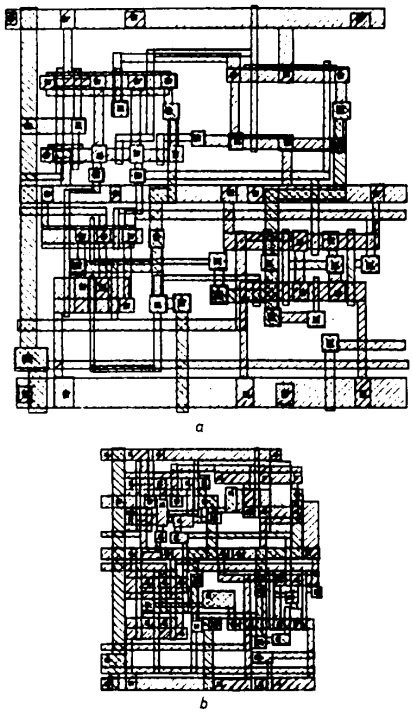
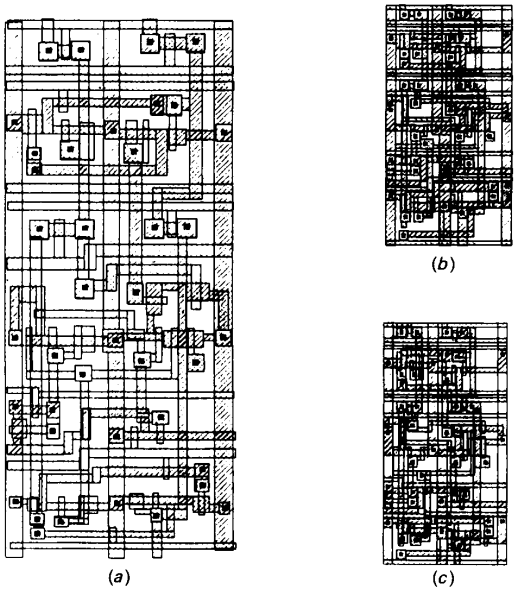


Fig. 15 *Experimental result*

a Source layout containing more than 20 transistors
b Compacted result with an area reduction of 67.5%



Compactor	Area	CPU	Source program for compaction	Design rule
SPARCS	123*198 = 24354	8	—	—
ZERRO	108*187 = 20196	378	—	—
EOC[14][24]	80.8*156.7 = 12661.4	46	~6000 lines	2.0 μm CMOS
This system	82.8*157.0 = 12999.6	~2100	~3000 lines	2.0 μm CMOS

(*d*)

Fig. 16 *Benchmark from Boyer's paper* [25]

a Source layout
b Compacted result by this expert compactor
c Compacted result by the algorithmic compactor presented in References [14 and 24]
d Performance comparison

deleted or modified without significantly affecting the modularisation of the whole set of expert rules. By refining better heuristics in the control strategy, and adding more rules to the domain knowledge base, we expect further improvement in both the runtime and compacted density. Besides, the main interest of our future extension will be to consider inducing the heuristic design techniques of a knowledge-based expert system to achieve a two-dimensional compaction scheme in a building block level.

8 Acknowledgment

Part of this work was supported by National Chiao Tung University and the National Science Council, Republic of China.

9 References

1 WU, C.F.E., WIJCIK, A.S., and NI, L.M.: 'A rule-based circuit representation for automated CMOS design & verification'. ACM/IEE Proc. of 24th Design Auto. Conf., 1987, pp. 786–792
2 CESEAR, T., IODICE, E., and TSAREFF, C.: 'PAMS: An expert system for parameterized module systhesis'. ACM/IEEE Proc. of 24th Design Auto. Conf., 1987, pp. 666–671
3 LUE, W.J., and McNAME, L.P.: 'PLAY: Pattern-based symbolic cell layout part I: Transistor placement'. ACM/IEEE Proc. of 24th Design Auto. Conf., 1987, pp. 659–665
4 CAI, H.: 'A rule-based global routing optimizer'. IEEE Proc. of ICCD, 1987, pp. 43–46
5 JOOBBANI, R.: 'An artificial intelligence approach to VLSI routing' (Kluwer Academic Publishers, 1986)
6 KOLLARITSCH, P.W., and WESTE, N.H.E.: 'TOPOLOGIZER: An expert system translator of transistor connectivity to symbolic cell layout', *IEEE J. of Solid State Circuits*, 1985, pp. 799–804
7 LIN, Y.L.S., and GAJSKI, D.D.: 'LES: A layout expert system', *IEEE Trans. on CAD/ICAS*, 1988, **7**, (8), pp. 868–876
8 KOWALSKI, T.J.: 'The VLSI design automation assistant: A knowledge based expert system'. CMU, Report #CMUCAD-84-29, 1984
9 WATANABE, H., and ACKLAND, B.: 'FLUTE: An expert floor planner full-custon VLSI design', *IEEE Mag. Design & Test*, 1987, pp. 32–41
10 KEDEM, G., and WATANABE, H.: 'Graph-optimization techniques for IC layout and compaction', *IEEE Trans. on CAD*, **3**, (1), 1984, pp. 68–105
11 CHOW, Y.E.: 'A subjective review of compaction'. ACM/IEEE Proc. of 22nd Design Auto. Conf., 1985, pp. 396–404
12 SOENHARDT, J., and LENGANER, T.: 'Algorithmic aspects of one dimensional layout compaction', *IEEE Trans. on CAD/ICS*, 1987, **6**, (5), pp. 903–910
13 NANDY, S.K., and PATNAIK, L.M.: 'Algorithm for incremental compaction of geometrical layout', *Int. J. of Comput. Aided Design*, 1987, **19**, (15), pp. 257–265
14 HSIAO, P.Y., and FENG, W.S.: 'An edge-oriented compaction scheme based on multiple storage quad tree'. IEEE Proc. of ISCAS, 1988, pp. 2435–2438
15 BROWNSTON, L., FARRELL, R., KANT, E., and MARTIN, N.: 'Programming expert system in OPS5' (Addison-Wesley Publishing Company Inc., 1985)
16 GIARRATANO, J.C.: 'OPS5 and OPS5+ basic guide'. AI Section, Johnson Space Center, 1986

17 WATERMAN, D.A.: 'A guide to expert systems' (Addison-Wesley Publishing Company Inc., 1986)
18 BIRMINGHAM, W., JOOBBANI, R., and KIM, J.: 'Knowledge-based expert systems and their application'. ACM/IEEE Proc. of 23rd Design Auto. Conf., 1986, pp. 531–539
19 BENTLAY, J.L., and FRIEDMAN, J.H.: 'Data structure for range searching', *Int. J. of Computing Surveys*, 1979, **11,** (4), pp. 397–409
20 ROSENBERG, J.B.: 'Geographical data structures compared: a study of data structures supporting region queries', *IEEE Trans.*, 1985, **CAD-4,** (1), pp. 53–67
21 NANDY, S.L., and RAMAKRISHNAN, I.V.: 'Dual quad-tree representation for VLSI designs'. ACM IEEE Proc. of 22nd Design Auto. Conf., 1985, pp. 663–666
22 SHAH, P.C., and MAHABALA, H.N.: 'A new compaction scheme based on compression ridge'. ACM/IEEE Proc. of 24th Design Auto. Conf., 1987, pp. 645–648
23 ACKLAND, B., and WESTE, N.: 'An automatic assembly tool for virtual grid symbolic layout'. Proc. of VLSI, 1983, pp. 457–466
24 HSIAO, P.Y., and FENG, W.S.: 'Using multiple storage quad tree on hierarchical VLSI compaction scheme', *IEE Trans. on CAD/ICS*, May 1990, **9,** (5), pp. 522–536
25 BOYER, D.G.: 'Symbolic compaction benchmarks — results'. IEEE Proc. of ICCD, 1987, pp. 209–217
26 HSIAO, P.Y., SYAU, CHEN YUNG, FENG, WU-SHIUNG, PARNG, T.M., and HSU, C.C.: 'A rule-based compactor for VLSI/CAD mask layout'. IEEE Proc. of Comput. Software Appl. Conf., 1988, pp. 35–42