論文 87-24-6-24

A Symbolic Layout Generator for CMOS Standard Cells Using Artificial Intelligence Approach

(인공지능 기법을 이용한 CMOS 표준셀의 심볼릭 레이아웃 발생기)

劉 宗 根*, 李 文 基*
(Jong-Keun You and Moon-Key Lee)

要 約

본 논문에서는 CMOS 표준셀의 심볼릭 레이아웃에 인공지능 기법의 적용방법을 제시하고 프로그램하였다. 셀 레이아웃은 트랜지스터의 배치작업과 배선작업으로 나누어 수행하도록 하였다. 배치작업에서는 휴리스틱 탐색기법을 이용하여 확산 영역의 분리를 최소화 하도록 하였고, 배선작업에서는 항상 올바른 해를 보장하기 위해서 먼저 거친 초기배선을 수행한다음 순방향 추론기법을 이용한 Rule-based 방식을 사용하여 레이아웃의 효율을 증가시키도록 하였다. 또한 입출력단자의 위치를 상하뿐만 아니라 좌우에도 가능하게 하여 보다 다양한 레이아웃을 얻을 수 있도록 하였다.

Abstract

Abstract-SLAGEN, a system for symbolic cell layout based on artificial intelligence approach, takes as input a transistor connection description of CMOS standard cells and environment information, and outputs a symbolic layout description. SLAGEN performs transistor grouping by a heuristic search method, in order to minimize the number of separations, and then performs group reordering and transistor reordering with an eye toward minimizing routing. Next, SLAGEN creates a rough initial routing in order to guarantee functionality and correctness, and then improves the initial routing by a rule-based approach.

I. Introduction

Since the VLSI design process includes various kinds of complicated and large scale problems, most of these problems are difficult to solve by conventional algorithmic approaches. In order to solve these problems, over the past decade, various kinds of heuristic algorithms have been developed and applied to real design problems. But the solutions by heuristic algorithms are still less satisfactory than an human expert's results [1].

However, recently, as new approaches to attempt to apply artificial intelligence strategies to VLSI design have been proposed, these problems seem to be solvable. Particulary, knowledge based system, a program in which the domain knowledge is explicit and separate from the program's other knowledge, have computers carry out expert-level design tasks by reasoning the given problem from domain

*正會員, 延世大學校 電子工學科
(Dept. of Elec. Eng., Yonsei Univ.)
接受日字 : 1987年 8月 22日
(※ This work was Supported by the Korea Research Foundation.)

knowledge [2], [3], [4].

According to this trend, this paper describes an artificial intelligence approach used for symbolic layout generation of CMOS standard cells [5], [6]. It also describes a program called SLAGEN (Symbolic Layout Generator). SLAGEN creates symbolic cells by separately performing transistor placement and routing. In performing placement, SLAGEN minimizes the number of separations by using a heuristic search method. A separation is required when there is no connection between physically adjacent transistors. In performing routing, SLAGEN uses a rule-based approach in order to apply human knowledge to routing. Also, it can produce a symoblic cell which satasfies the requested position of each input/output signal and the height of the cell specified by a designer. SLAGEN was implemented in the GCLISP (Golden Common Lisp) [7] and runs on a personal computer such as IBM-AT.

II. System Overview

SLAGEN requires two inputs: MODEL and ENVIRONMENT. First, a given circuit must be defined in terms of transistor connectivity. Second, it requires information about the position of each external signal and the height of the cell. SLAGEN outputs symbolic layout descriptions in ICDL (Intermediate Circuit Description Language) [8]. ICDL describes the circuit of a cell symbolically in terms of transistors, wires and interlayer contacts arranged on a vitual grid. SLAGEN consists of two subsystems, TRANPLA and TRANROUT (see Fig. 1). The former performs transistor placement and the latter performs routing.

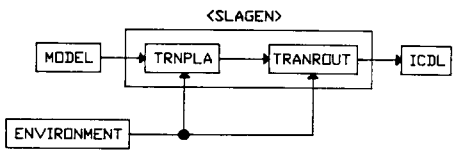


Fig. 1. The architecture of SLAGEN.

III. Tranpla

In order to reduce cell area, the transistors must be so placed that the number of separations is minimized and the given environment conditions are satisfied. TRANPLA executes transistor placement by three steps: transistor grouping, group reordering and transistor reordering.

1. Transistor grouping by a heuristic search method

In CMOS, all complementray static gates may be designed using a single row of n-transistors below a single row of p-transistors, aligned at common gate connections. Therefore, transistor placement is a matter of determining the order of the TRPAIRs so that the number of separations may be minimized. A TRPAIR consists of a n-transistor and a p-transistor aligned at a common gate connection.

The role of transistor grouping is to organize all the TRPAIRs of a given circuit into several groups or a group so that the TRPAIRs of each group may share a common source-drain connection. Transistor grouping is executed by a heuristic search method, one of the problem solving methods of AI [9]. The objective of this procedure is to minimize the number of groups because the number of groups corresponds to the number of separations and so, the smaller the number of groups is, the smaller becomes the cell area.

This search problem is characterized by an initial state and a goal state description. An operator transforms a state into another state. The objective is to find a goal state.

(1) state description

A state is described by a list of groups including TRPAIRs. Each group is expressed as a list of TRPAIRs which can share a common source-drain connection, but group 0, a initial group, includes TRPAIRs ordered randomly.

(2) *initial state*

A initial state is defined as the following. Its group 0 includes all the TRPAIRs of a given circuit except one and its group 1 has a TRPAIR group 0.

(3) *operator*

The operator tries to search the group 0 of a given state for the TRPAIR which can share a common source-drain connection with the TRPAIRs included in the current group. The current group indicates the group which

has been created last. For example, in the case of the initial state, the current group is the group 1. If such a TRPAIR exists in the group then it is moved from the group 0 to the current group. Otherwise, a new group is created and a TRPAIR selected randomly from the group 0 for the operator to create several new states from a state because several satisfied TRPAIRs can be found.

(4) *goal state*

The goal state is when the group 0 of the state is empty. This means that all of the TRPAIRs have been used for transistor grouping.

With the above definitions, the search process is executed according to the search flow for transistor grouping shown in Fig. 2. In the search flow, a node is corresponding to a state and the root node is the initial state. CURRENT has a node to be expanded next.

We use an evaluation function to order nodes on OPEN and to select a node to be expanded next. The value of the evaluation function of a node is equal to the sum of two terms, the number of groups and the number of TRPAIRs in group 0 The lower the function value of a node is, the nearer the node will approach a goal node. Therefore, among the nodes on OPEN, the node having the lowest function value is selected and then expanded. And so, it is possible to place the transistors so as to minimize the number of separations.

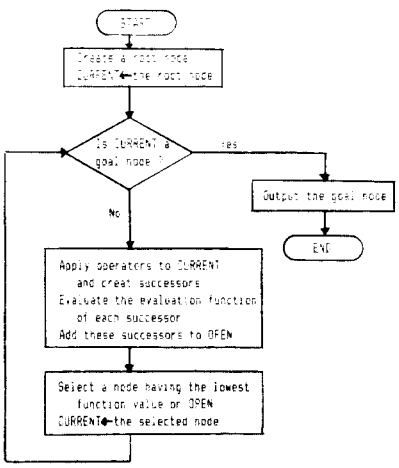


Fig. 2. Search flowchart for transistor grouping.

2. Group reordering and transistor reordering

Since, in the transistor grouping process, environment information is ignored and only minimizing the number of separations is considered, it is necessary to reorder the groups and the TRPAIRs of each group according to environment information, in order to reduce the complexity of routing. Fig. 3 illustrates an example of group reordering and an example of transistor reordering is shown in Fig.4. The role of group reordering is to reorder the groups according to environment conditions. The role of transistor reordering is to reorder the TRPAIRs of each group, if needed, but new separations must not be created. As noted above, TRANPLA minimizes the number of separations and executes transistor placement with an eye toward minimizing routing.

Fig. 3. An example of group reordering.

Fig. 4. An example of transistor reordering.

IV. Tranrout

After completing the placement, SLAGEN invokes TRANROUT. TRANROUT acts on the placement description, the output of TRANPLA, together with environment information. TRANROUT creates a rough initial routing in order to guarantee functionality and correctness but it is indeed very wasteful in area and performance. Therfore, in order to improve the initial routing, TRANROUT uses a rule-based approach [10]. TRANROUT operates by examining the rough layout and iteratively improves the routing according to a set of rules.

1. Initial routing

Using the virtual grid, the initial router interconnects between the same signals by using only one wire at one tract. And the metal, diffusion and polysilicon layers can be used in routing. The initial routing consists of the following five steps.

STEP1) placement of pins
STEP2) placement of power lines
STEP3) placement of transistors
STEP4) routing of gate signals
STEP5) routing of drain/source signals

2. Rule-based system

The rule-based system acts on the initial rough routing to create improved results in the form of ICDL. In other words, this system examines the rough routed layout and improves it by successive application of the rules. As Fig. 5 shows, this system consists of three major components: a data memory, a rule memory and a inference engine.

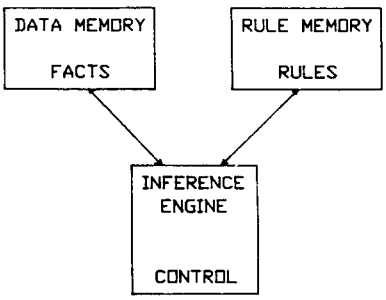


Fig.5 The architecture of the rule-based system.

1) data memory

A data memory serves as a global database of symbols representing the symbolic layout of a given cell. Basic elements, such as devices, wires, pins and contacts, are required to represent the symbolic layout of a cell. Therefore, these symbols are stored in the data memory, and also have attributes and values. Attributes are properties associated with the symbols corresponding to layout elements. Table 1 shows the attributes of each elements. Values specifies the specific nature of an attribute in a particular situation.

Table 1. Attributes of each layout element.

LAYOUT ELEMENT	OBJECT NAME	ATTRIBUTE
WIRE	WIR	TYPE, B-POS, E-POS, B-MARK, E-MARK NAME
CONTACT	CON	POS, CUT1, CUT2, NAME
PIN	PIN	TYPE, POS, NAME
DEVICE	DEV	TYPE, POS, GAT, L-DIF, R-DIF, NAME

2) rule memory

The rules that embody the human knowledge about routing are stored in rule memory. The number of routing rules that this system uses at present are 32. Fig. 6 illustrates five of the 32 routing rules.

Each rule has a condition part, indicated by the keyword "IF", which describes the data configuration for which the rule is appropriate. A rule also has an action part, indicated by the keyword "THEN", which gives instructions for changing the data configuration. The method used to test against the contents of data memory is conditon function calls. Also, the method of function calls is used for right-hand-side actions of each rule. Fig. 7 shows the form of the rule which is corresponding to type 1 of Fig. 6.

3) inference engine

An inference engine is needed to execute the rules. This system uses the forward-chaining strategy which matches the left-hand sides of rules to update the knowledge base by making changes to data memory. Also, rule grouping and data filtering are used in order to improve the performance of the system.

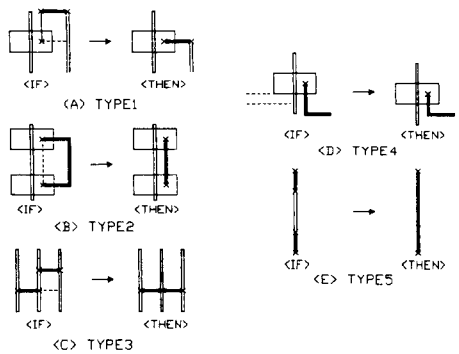


Fig. 6. Examples of Routing Rules.

```
(SETQ R-RULE13 (ROUTING-PDIF3
      (IF     (PROPERTY %WIR1 NAME %NAME1)
              (FIND-SECOND-MARK (%NAME1) %CON1)
              (NEXT-CUT (%CON1 %WIR1) %WIR2)
              (NEXT-MARK (%WIR2 %CON1) %CON2)
              (NEXT-CUT (%CON2 %WIR2) %WIR3)
              (TEST-WIR-TYPE (%WIR3 ( POLY)))
              (FIND-FIRST-POS (%WIR1) %POS1)
              (PROPERTY %CON2 POS %POS2)
              (TEST-FORM-PDIF3 (%POS2 %WIR3))
              (XY-PAIR (%POS2 %POS1) %POS3)
              (TEST-DIF-TRACKS (%POS1 %POS3 ( PDIF)
             %NAME1) %POS4 %POS5))
      (THEN (MOVE-CON (%CON1 %POS4)
              (MOVE-CON (% CON2 %POS5))
              (MOVE-WIR (%WIR2 %POS4 %POS5))
              (WIR-SHORTEN (%WIR3 %POS2 %POS5))
              (REMOVE-WIR (%WIR1))
              (REARR-CON (%CON1))
              (REARR-CON (%CON2)))))
```

Fig. 7. A diffusion routing rule in the GCLISP.

The rules which are in the conflict state with each other, are organized into a group and are executed through the recognize-act cycle shown in Fig. 8. This cycle consists of three parts; rule matcher, rule selector and rule executor. Rule matcher finds all of the rules that are satisfied by the current contents of data memory. Rule selector applies a conflict resolution strategy to determine which rules will actually be executed, Rule executer executes the selected rule. This cycle stops when rule matcher can find no matches. The other rules which are some what independent have priority each. And, rule matcher begins from the rule having the highest priority.

And, this system uses a data filtering strategy, which reduces the number of data elements that are matched against each rule by each rule.

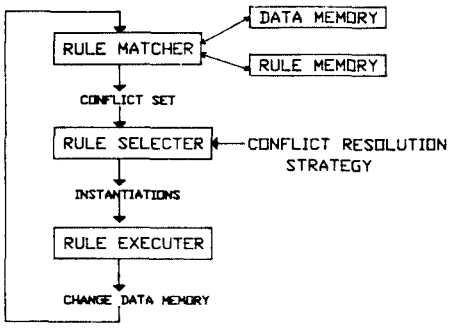
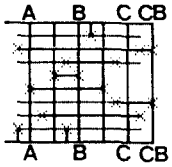


Fig. 8. Recognize-act cycle.

V. Examples and discussion

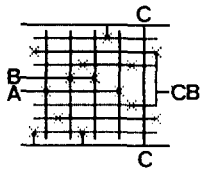
Fig. 9 displays SLAGEN 's results for the same carry cell with differently requested environment inputs. In Fig. 9a I/O ports of of the carry cell are at the top and bottom, and in Fig. 9b. I/O ports are at the left and right. And, Fig. 9c shows the result having the differently requested height from that in Fig. 9a or Fig. 9b. Fig. 10 displays the initial routing and the final result of dlatch cell, and Fig. 11 displays the result of jk flip-flop.

```
(SETQ ENVIRONMENT
      (CELL-HEIGHT= D
       NDIF-HEIGHT= D
       PDIF-HEIGHT= D
       (A    POS= TB)
       (B    POS= TB)
       (C    POS= TB)
       (CB   POS= TBR)))
```



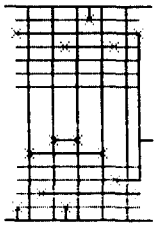
(a) result with IO Ports at the Top & Bottom.

```
(SETQ ENVIRONMENT
      (CELL-HEIGHT= D
       NDIF-HEIGHT= 3
       PDIF-HEIGHT= 3
       (A    POS= L  FPOS= 4)
       (B    POS= L  FPOS= 5)
       (CB   POS= R  FPOS= 4)))
```



(b) result with I/O Ports at the left & Right.

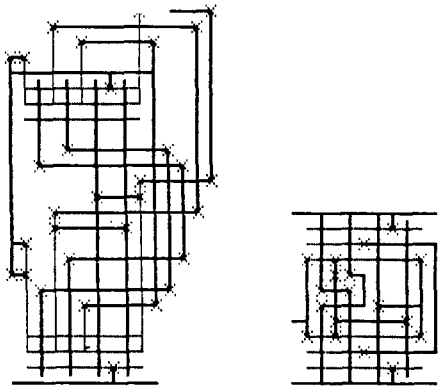
```
(SETQ ENVIRONMENT
      (CELL-HEIGHT= 17
       NDIF-HEIGHT= 4
       PDIF-HEIGHT= 6
       (A    POS= TB)
       (B    POS= TB)
       (C    POS= TB)
       (CB   POS= R)))
```



(c) result with Cell Height of 17 Tracks.

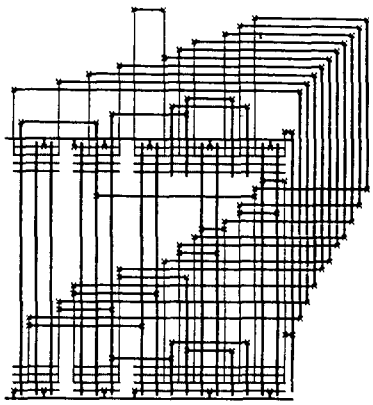
Fig.9. Carry cells created by SLAGEN with various environment inputs.

As shown in Fig. 9, SLAGEN can create various layouts for the same circuit by changing the environment input. And, as shown in Fig. 10 and Fig. 11, SLAGEN minimizes the number of separations by the heuristic search method. Of methods to minimize the number of separations, there is the minimal interlace algorithm proposed by T. Uehara W. Vancleemput[11]. This algorithm is not

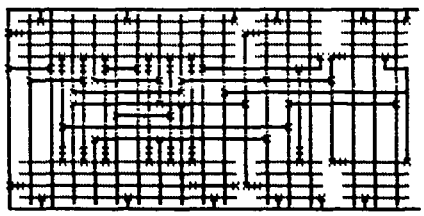


(a) Initial rough layout (b) Final result

Fig. 10. Symbolic layout of the DLATCH cell by SLAGEN.



(a) Initial rough layout



(b) Final result

Fig. 11. Symbolic layout of the JK flip-flop cell by SLAGEN.

Vancleemput [11]. This algorithm is not applicable to circuits including transmission gates, but SLAGEN is applicable to all kinds of circuits. And, several cell generators don't deal withcircuitsincludingtransmission gates [2],[12] or have disadvantage that the layout area increases because of transmission gates [13]. However, SLAGEN performs compact layout for circuits including transmission gates.

VI. Conclusion

This paper proposes the application of artificial intelligence strategies to symbolic layout generation of CMOS standard cells. By the heuristic search method, the number of separations can be minimized. By the rule-based approach in routing, human knowledge about routing can be employed easily. And, since SLAGEN can creat layouts having the ports in the positions which are requested,by designer, we can get more varieties in layouts.

In order to improve the performance of SLAGEN, it is needed to improve SLAGEN's run-time efficiency and control strategy, and more powerful rules are reqired. And, the next research phase will focus on converting symbolic descriptions to mask level descriptions.

References

[1] S. Goto, "Design methodologies", *Advanced in CAD for VLSI,* vol. 6, pp. 441-464, 1986.

[2] Pall. W. Kollaritsch and N. Weste, "A rule-based symbolic layout expert", *VLSI Design,* pp. 62-66, August 1984.

[3] Rostam Joobbani and Daniel P. Siewiorek, "WEAVER; A knowledge based routing expert", *IEEE Design & Test,* pp. 12-23, Feb. 1986.

[4] Hugo J. De man, "Dialog; an expert debugging system for MOS VLSI design," *IEEE Transaction on CAD, CAD-4,* no. 3. pp. 303-311, July, 1985.

[5] J.K. You, "A symbolic layout system for CMOS standard dells using artificial intelligence approach," M.S. thesis, Yonsei university, Department of Electronic Engineering, 1987.

[6] J.K. you, B.Y. Kim, M.K. Lee, "An auto generation of symbolic layout for CMOS standard cells using artificial intelligenece approach", *The 40th Aniversary KIEE*

Autumn Conf. Proceeding, vol. 9, no. 2, Dec., 1986.

[7] "Golden Common Lisp", Gold Hill Computer, Inc.

[8] Neil Weste and Kamran Eshraghian, "Principales of CMOS VLSI design", *A System Perspectives,* Addison Wesley, pp. 271-308, 1985.

[9] Nils J. Nilsson, "Principles of artificial intelligence", Springer-Verlag Berlin Heidlberg New York, pp. 53-96. 1982.

[10] Paul Harmon, David King, *"Expert system"*, wiley press, pp. 34-60, 1985.

[11] Takeao Uehara and William M. Vancleemput, "Optimal layout of CMOS functional array", *IEEE Trans. on Computers,* vol. C-30, no. 5, pp. 305-312, May, 1981.

[12] Nam Tosuntikool and Charles L. Saxe, "Automated design of standard cells", *IEEE Custom IC Circuits Conf.,* pp. 110-114, 1984.

[13] H.H. Kim, "Auto Generation of CMOS Standard Cells," M.S. Thesis KAIST, Department of Electronic Engineering, 1986.

Acknowledgement

The authors gratefully acknowledge the support of the Korea Research Foundation.