TM-0275

# AN EVALUATION OF PARALLEL LAYOUT ALGORITHM

by

Y. AOYAGI, K. YOSHINAGA

and N. SHIRAKI

OKI ELECTRIC INDUSTRY CO.,

February, 1987

# AN EVALUATION OF PARALLEL LAYOUT ALGORITHM

YOSUKE AOYAGI, KAZUHIRO YOSHINAGA, NOBORU SHIRAKI

VLSI R & D CENTER, OKI ELECTRIC INDUSTRY CO., LTD.

550-1, HIGASHIASAKAWA-CHO, HACHIOJI-SHI, TOKYO 193 JAPAN

## ABSTRACT

For coping with VLSI function diversification and greater integration, knowledge engineering and parallel processing for high speed processing has become more critical.

This paper reports that the parallel layout extracted by a partition layout algorithm [1] has been used for the polycell layout of a standard cell LSI. The algorithm description language used for this purpose is GHC (Guarded Horn Clauses) [2], which will be used as the kernel programming language in the fifth-generation computer project.

## 1. INTRODUCTION

### 1.1 Purpose

The future CAD system architecture allows two approaches. In one, a parallel processing algorithm is employed for high-speed processing and the conventional CAD software algorithm is replaced with a hardware algorithm. In the other, an integrated intelligent CAD system is constructed using a knowledge base (KB) and inference engine to cope with the large amount of information and know-how on VLSI designs and rapid VLSI technology diversification.

This study is for the latter CAD system architecture. It covers basic components for intelligent CAD systems that run using the knowledge base and inference engine on the high-parallel machine to be developed by the fifth-generation computer project.

### 1.2 Guarded Horn Clauses (GHC)

GHC is a logic language that can describe parallelism explicitly. It is the main (kernel) language (machine word in coventional computers) that can describe knowledge-based or intelligent processing systems or application programs. See Figure 1.1 for details.

The basic elements of a GHC program include the process and communication. A process is the basic unit for data processing in parallel calculation. As soon as the necessary data has been obtained, calculation starts automatically. When a data shortage occurs, the synchronization queue is used to wait for data to arrive. In addition, the internal status can be held or updated as required. Communication is a basic means of data exchange between processes. It is realized by a shared variable. Communication can take place on 1:1 or 1:N basis. And in many cases, it is handled as a message stream.

The entire GHC program can be thought of as a network of processes that communicate with each other. See Figure 1.2 for details.

### 1.3 Application for a CAD System

For modeling a CAD system, a standard cell LSI layout problem is selected for this paper. This is because the problem is on the boundary between the logic design and device design (problem solving is difficult because of the restrictions of both designs), the search space is extremely large, and a high-speed intelligent search is critical.

A bipartition layout algorithm is used for two reasons. The first is that process network modeling is easy because the structure is simple. The second is that high-parallel execution can be realized because each partition enables parallel execution. In addition, as a first prototype, the evaluation function is used only for logic line length.

## 2. LAYOUT MODEL [3]

A layout model comprises both strategy information based on information and know-how (prepared by designers) and layout information extracted from the CAD data base. It has a structure of GHC streams. The program receives it as input data. In representing a layout model, information required for each process is stored in a compact format as internal information as much as possible to maintain process independence and improve the execution efficiency when the layout model has been instantiated as a group of parallel processes during program execution.

Figure 2.1 shows an example of strategy information. This information is obtained by modeling the plan based on designers' know-how. There is only one model for the corresponding block. It is stored in one stream in group unit (one page of a circuit diagram or unit specified by the designer).

Figure 2.2 shows an example of layout information. This layout information is obtained by the logic design data and cell library being modeled according to the layout plan. It is provided for each group in the block. Layout information for one group is stored in one stream in cell unit.

## 3. LAYOUT SYSTEM

### 3.1 Configuration

Figure 3.1 shows a system configuration. This system consists of a clustering section (group processes for each group layout), layout section (solver processes for cell layout in each group), and interface section (manager, output, and display processes for external I/O processing).

### 3.2 Staged Generation and Test Method [4], [5]

Because a layout problem involves an extremely large search space, solutions are generated statistically or dynamically through some stages to improve the searching efficiency and so that the stages can be executed in parallel for a higher-processing speed (See Fig. 3.2).

The manager process generates each group process statistically (non-variable process network) from input data (layout model) using independent AND parallel nodes, then it stores the layout result of each group. Each group process generates the top level of the solver process from the received data and collects the layout result for each cell in the group. The solver process generates two partitions for a layout pattern and selects the appropriate two partitions through the evaluation function. Then, it generates a solver recursively and dynamically (variable process network) as an independent AND parallel process (2 branches) for each partition. When the number of partition elements reaches 1, each solver assumes that a cell layout has been completed and outputs the layout result.

When a dependent AND parallel execution system (inter-process communication outside the binary tree procedure network) is used, more detailed control and data exchange can be performed between the processes although the model becomes more complex and the communication cost increases. In this study, an independent AND parallel execution system is used because the model is simple and the communication cost is lower.

## 3.3 Solver

A solver consists of a selector, an exhaustive solver, and a heuristic solver (See Fig. 3.3). The selector receives a cell list and checks the number of cells (search space size). When the number of cells is smaller than the selection value, the selector generates an exhaustive solver. This is modeled by all solutions search type programming in GHC and executed in the OR parallel mode. And when the number is greater than the selection value, the selector generates a heuristic solver. This is modeled by the serial exchange algorithm, and consequently the degree of parallelism is low. This selection value must be properly determined in advance, considering the processing efficiency of the machine and GHC processor. If a greater selection value is specified, a better result can be expected but the excution speed is lowered.

## 4. EVALUATION

### 4.1 Layout Result

Figure 4.1 shows the result of applying the layout result of a block having 78 cells, 65 terminals, and 6 groups to an existing routing system. While, Table 4.1 shows a comparison between the routed result of the above method and that of manual layout.

In the layout by this system, the logical wire length is increased by approx. 30% and the connection rate decreased by approx. 5% as compared with the case of manual layout. Now, detailed studies are under way to analyze its cause.

## 4.2 High-speed Processing by Parallel Machines

This is modeled by a basically independent parallel AND execution system. The communication traffic between processes, therefore, is considered to be relatively small mainly the input of data and output of results; data is not exchanged between processes often).

For example, suppose parallel machines consist of tens of host processors (H.P.), each made up of tens of processor elements (P.E.) (Fig. 4.2). Improving the processing speed tens times may be possible by the elaborate planning of the static allocation of group processes to H.P. and the dynamic allocation of solver processes to P.E.

## 4.3 Guarded Horn Clauses (GHC)

Parallelism is extracted by the bipartition layout algorithm. It is modeled by a binary tree process network and described in GHC. Because GHC can describe parallelism (having a synchronization mechanism between parallel processes) explicitly and the bipartition algorithm is relatively simple, model description is relatively simple.

The current GHC processor is the C-Prolog interpreter that runs on the VAX8300 computer. Its processing speed is slow and a large memory is required. High-speed parallel machines and knowledge processors more sophisticated than GHC will be required in future.

## ACKNOWLEDGEMENTS

This study is one of the operations to be performed by the fifth-generation computer project. Acknowledgements are due to Mr. Furukawa (ICOT Laboratory Vice-manager), Mr. Ito (First Office Chief), and ICOT members for reviews.

## REFERENCES

[1] David P, La Potin and Stephen W. Director.
[1986] Mason: A Global Floorplanning Approach for VLSI Design. IEEE Transactions on Computer Aided Design, Vol. CAD-5, No.4, pp477-490.

[2] Ueda. K.
[1985] Guarded Horn Clauses. ICOT Tech. Report TR-103, Institute for New Generation Computer Technology, Tokyo.

[3] Hamazaki. R, Shiraki. N and Hirakawa. K.
[1986] Cell Placement Expert System K/cp For VLSI Design. IPSJ, DA-32-2.

[4] Ohki. M.
[1985] Search Programing in KL1. ICOT Tech. Memorandum TM-149, Institute for New Generation Computer Technology, Tokyo.

[5] Ueda. K.
[1986] Making Exhaustive Search Programs Deterministic. In Proc. 3rd International Conference on Logic Programing and Shapiro. E. (ED.), Lecture Note In Computer Science 225, Springer-Verlag pp 270-282.

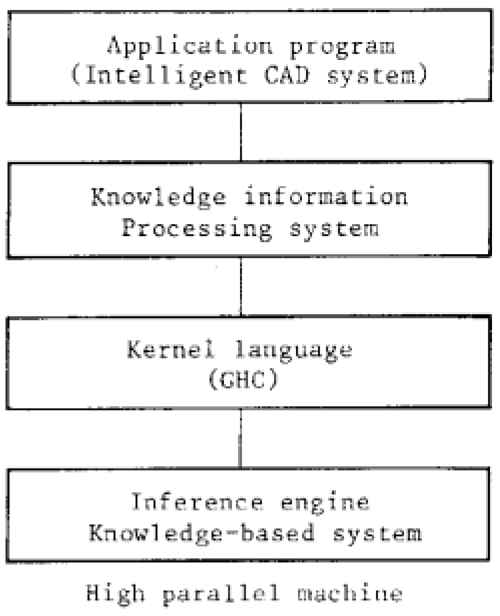


Fig. 1.1 Fifth Generation Computer

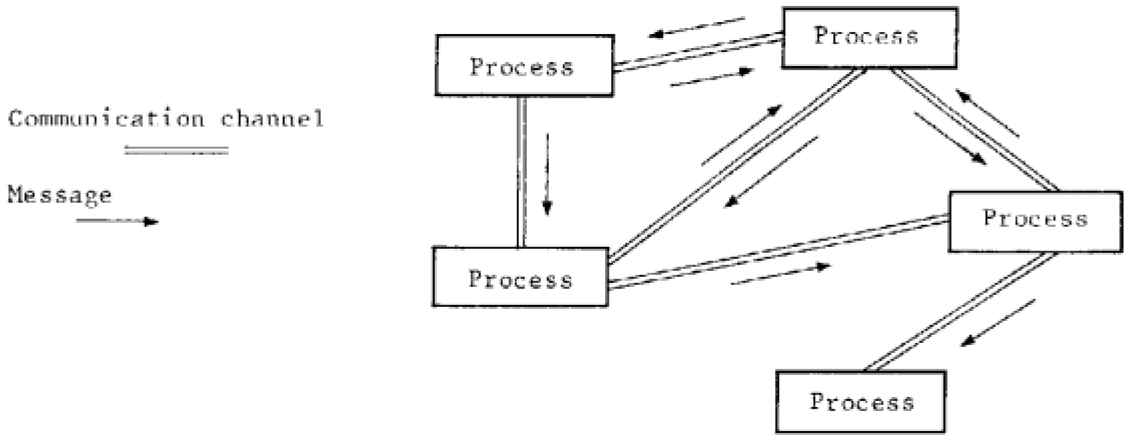
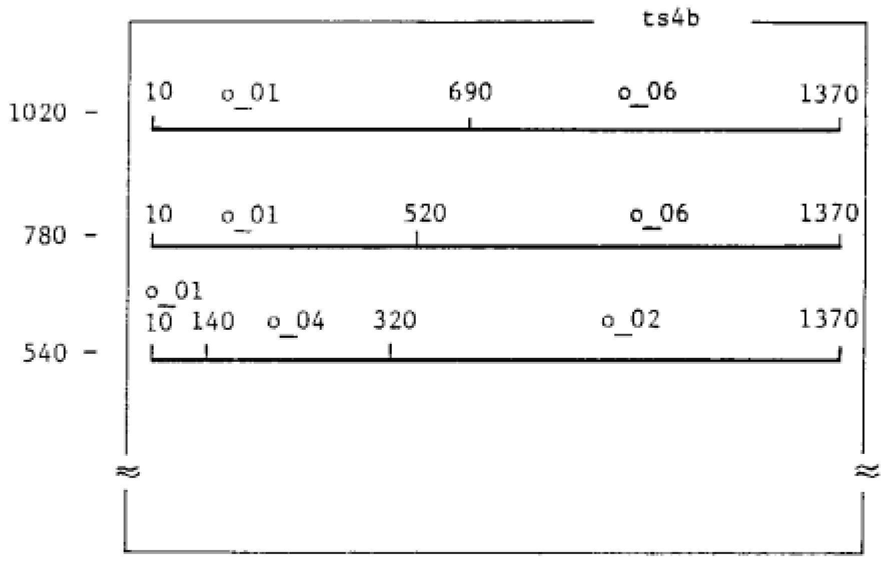
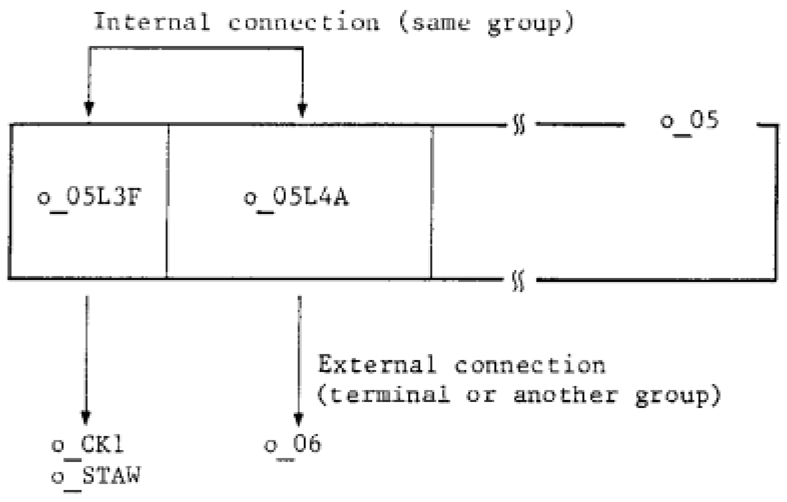


Fig. 1.2 GHC Programing



```
         Block name
                    Output stream
block (ts4b, E11): - true !
   E11 = [               Group name
          grp (o_06,                X axis
               [
                 cha (o_06, 780, 520, 1370),  } Area
                 cha (o_06, 1020, 690, 1370)  } information
               ]),
          grp (o_05,            Y axis
               [
                 cha (o_05, 300, 610, 1370)
               ]),
          grp (o_04,
               [
                 cha (o_04, 60, 10, 320),
                 cha (o_04, 300, 10, 610),
                 cha (o_04, 540, 140, 510)
               ]),
```

Fig. 2.1 An Example of Strategy Information



```
       Group name
                     Output Stream    Library
                                      reference name
group (o_05, E11): - true !
   E11 = [                                   Width
            ell (o_05L3F, o_D2ND, 80,
                    [
Cell name             con (o_nil, o_CK1, 940, -5, 1),      } External
                      con (o_nil, o_STAW, 1010, -5, 1)     } connections
                    ],
                    [
                      con (o_05, o_05L4A, 1),
                      con (o_05, o_05L4C, 1),              } Internal
                      con (o_05, o_05L4D, 1),              } connections
                      con (o_05, o_05L4E, 1)
                    ]),                       Connection ratio
   ell (o_05L4A, o_DFN, 170,
                    [
                      con (o_06, o_06, 987, 900, 5)
                    ],
                    []),                 Position
   ell (o_05L4C, o_DFN, 170,
                    [
                      con (o_06, o_06, 987, 900, 5)
```

Fig. 2.2  An Example of Layout Information

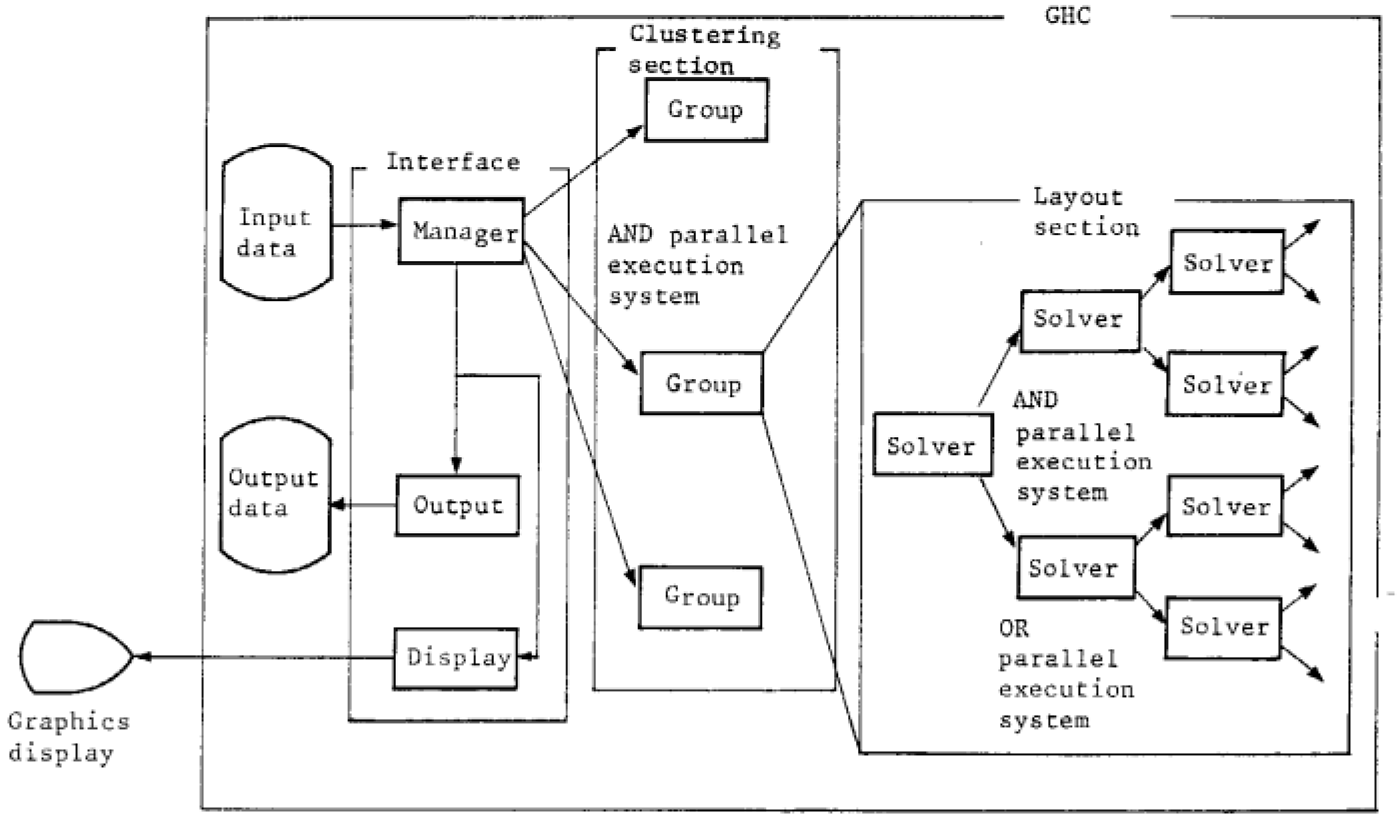


Fig. 3.1  System Configuration

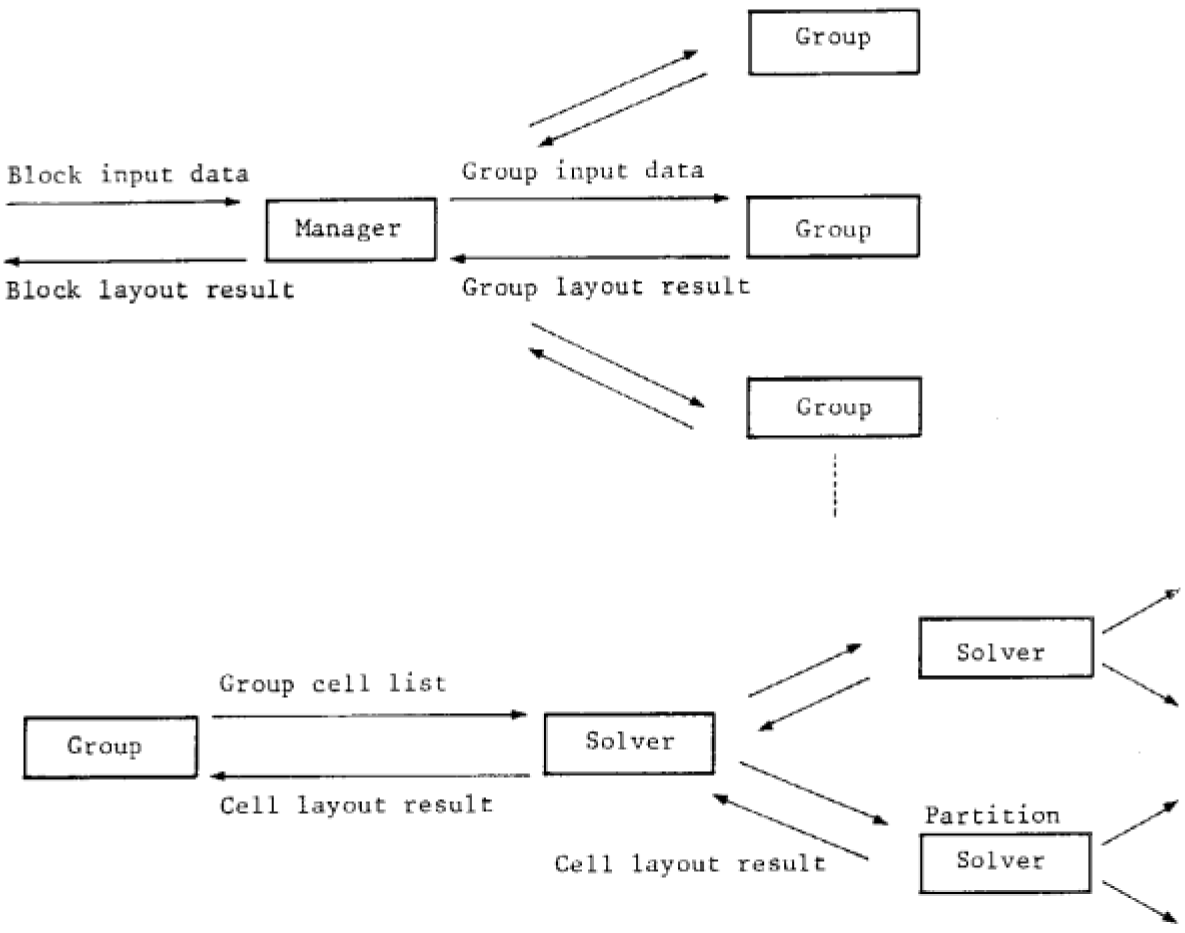


Fig. 3.2  Applied Generate and Test Method

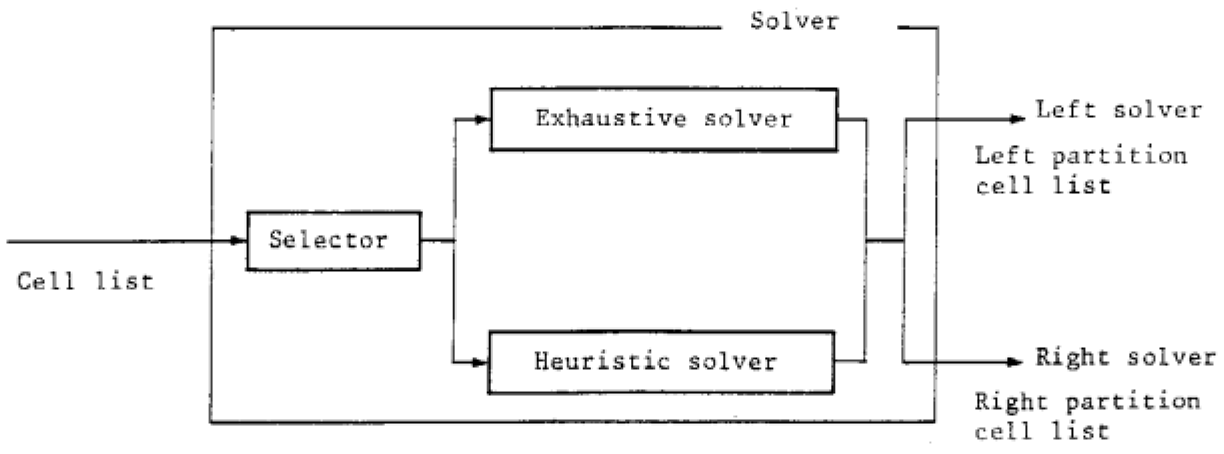


Fig. 3.3  Solver

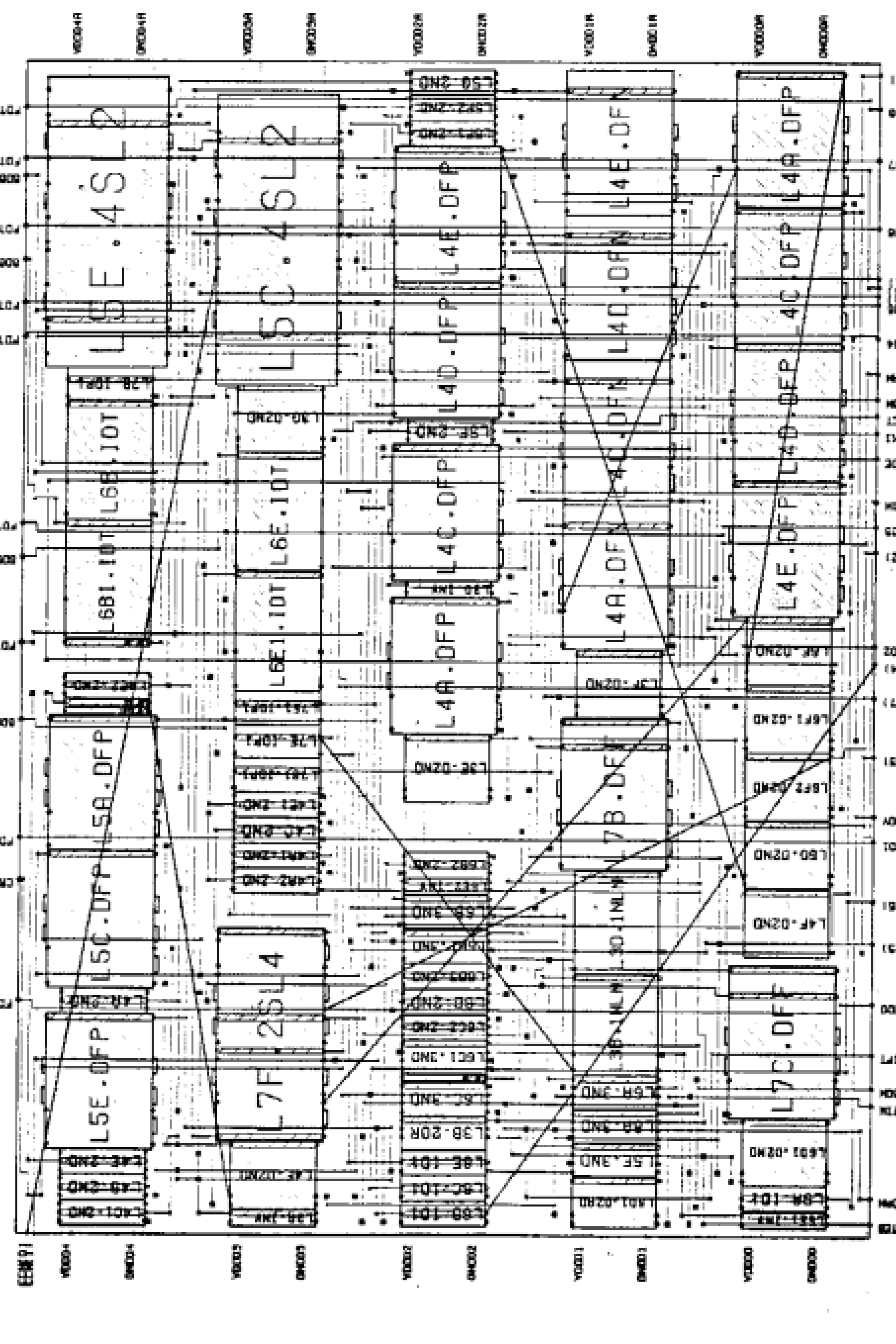


Fig. 4.1 Result of Layout Wiring

Table 4.1 Comparison with Manual Layout

| | | |
|---|---|---|
| Number of cells | 78 | |
| Number of terminals | 65 | |
| Number of groups | 6 | |
| Method of layout | Manual | Program |
| Sections not yet connected | 1 | 10 |
| Connection rate (%) | 99.5 | 94.7 |
| Logical wire length | 29797 | 39508 |
| Physical wire length | 34303 | 41295 |
| CPU (sec) | - | 1629.8 |
| Memory (MB) | - | 28.3 |

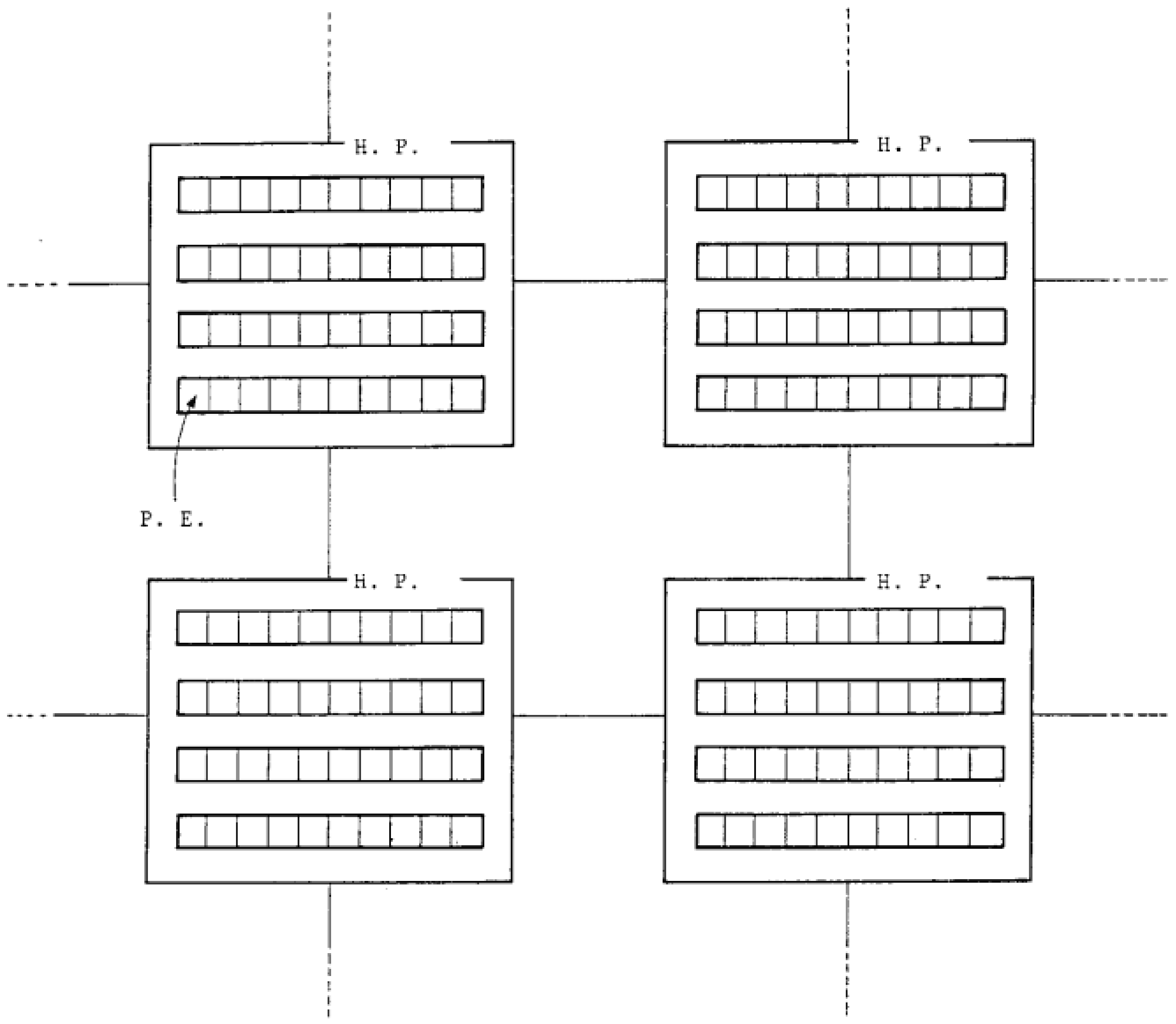


Fig. 4.2 An Example of Parallel Machine