TR-0771

Co-HLEX: Co-operative Recursive LSI Layout Problem Solver on Japan's Fifth Generation Parallel Inference Machine

by

T. Watanabe & K. Komatsu (Hitachi)

April, 1992

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome

(03)3456-3191~5
Telex ICOT J32964
Minato-ku Tokyo 108 Japan

Institute for New Generation Computer Technology

Co-HLEX: Co-operative Recursive LSI Layout Problem Solver on Japan's Fifth Generation Parallel Inference Machine

Toshinori Watanabe and Keiko Komatsu

Systems Development Laboratory, Hitachi, Ltd.
1099 OHZENJI, ASAO-KU, KAWASAKI-SHI, KANAGAWA, 215 JAPAN
Tel: (044) 966-9111, Telex: 3842-577, Fax: (044) 966-6862

Abstract

Co-HLEX is a co-operative hierarchical layout problem solver developed as an application program of parallel inference machines; Multi-PSI and PIM. The kernel of Co-HLEX is a hierarchical recursive concurrent theorem prover nicknamed HRCTL. Due to its recursive nature, HRCTL has a size of only O(1,000) lines in KL1; the kernel language of ICOT. Due to its stream-parallel and distributed-memory architecture, nearly linear time complexity could be attained. Moreover, shape and wire abutment among modules running in parallel could be made possible through message passing co-operation. In this paper, a brief overview of Co-HLEX is given with its application to bipolar-analog LSI layout.

1 Introduction

The main role of Co-HLEX development in Fifth Generation Computer System project was to find some answers to the following questions;
(Q1) Are there any real-world problems which require symbolic, parallel problem solving?
(Q2) If they exist, can we find any new parallel algorithms to solve them?
(Q3) Can we find elegant descriptions of these algorithms?
(Q4) Can we execute these descriptions effectively on Multi-PSI or PIM?
(Q5) What are the new break-throughs brought about?
(Q6) What are the new problems to be pursued further?
We picked up an LSI layout problem due to the following reasons;
(R1) It is and will be one of the gigantic real-world problems requiring massive computation power. At present, the number of rectangles contained in the layout of 1 cm^2 DRAM chip is almost equal to that of 100 m^2 rectangles covering Japan. New ideas including parallel computation are greatly required to cope with this complexity.
(R2) Both development and enhancement of a layout system, with more than 1 million-lined program codes, consume huge amount of programmers' unrewarded labour. The possibility of more elegant program descriptions should be investigated.
One of our ideas is the use of the recursion principle to reduce it [Kleene 1952]. The other is the use of a streamed parallel process network computation model to give an elegant description of mutually related layout objects. We nicknamed our algorithm HRCTL (Hierarchical Recursive Concurrent Theorem prover for Layout). The kernel language KL1, which runs on Multi-PSI and PIM, can be a powerful tool to implement them.
(R3) The classical divide and conquer - the hierarchical problem solving - works well as long as subproblems correlate weakly. In case of LSI layout, this premise cannot be sufficed. Neighbouring modules are not independent in that they should have abutted shapes and wires to avoid dead spaces. Our idea is the use of communication among modules to solve this AND typed dependency.

In the following, Section 2, 3, 4, and 5 give basic concepts, an overview of Co-HLEX, a brief complexity consideration, and experiment results, respectively. Based on them, we hope to assert that parallel symbolic computation can contribute much to LSI design automation.

2 Basic Concepts

2.1 Layout Problem and Solution Representation

The original layout problem which Co-HLEX solves can be specified by a Prolog goal:

```
:- mode solve_a_layoutproblem(+,-,+,+).
?- solve_a_layoutproblem(CirNet, LPlan, Proc, Constr).
```

where arguments have the follwing meanings as shown in Figure 2.1.

CirNet::= A circuit network represented by modules and module connection nets.
LPlan::= [PQtree, Wires].
PQtree::= A quadtree [Samet 1984, Otten 1982, Luk et al. 1986] representing a placement by a slice hierarchy each node of which carries a module name placed in the slice and peripheral connector names placed on north-, west-, south-, and east-edge of the slice.
Wires::= [[Conns, Lines] | Wires].
Conns::= Set of peripheral- and inner-connectors of a net. Note that induced connectors are peripheral connectors of subnodes. Inner connectors include terminal points and vias. Vias are through-holes connecting different silicon layers on the chip. An induced connector is introduced at

each point where a wire crossses a slice edge.

Lines::= Set of line segments spanning two connectors of a net. It includes connector names on both ends, the run layer name, and the line width.

Proc::= LSI fabrication process name which affects geometrical shapes of modules, usable wiring layers, line width, minimum allowances among objects, and others.

Constr::= A list of constraints including a set of proximity conditions of modules, usually called "Pairs", the topmost PL (planned layout - a planned chip size; Width and Height, and a set of planned peripheral connector placements).

2.2 Recursive Problem Solving

The slicing structure representation of a layout permits us the following recursive problem solving.

(S1) If the module is an indivisible leaf cell, import its layout from a library. If the module is a divisible block, divide the original layout problem, composed of a circuit data and a PL (planned layout), into at most 4 subproblems, each having a homologous structure with the original problem.

(S2) Solve all the subproblems in parallel using the recursion principle. If much processing elements or PEs are available, fork these subproblems on different PEs.

(S3) Aggregate finished layouts of subproblems following the placement plan given in (S1) to generate the layout of the original problem.

2.3 Problems and Solutions

(SL1) Pre-compilation of circuit net into a quadtree

In (S1) above, the original planned area should be divided into at most four slices, the circuit should also be divided into relevant number of subcircuits, and their embedding plan into slices should be made. To avoid the untractable computational complexity caused by these two divisions and one embed, the CirNet is transformed into a quadtree-shaped process network named CMPN (Circuit Module Process Network) before layout generation. Each node in the CMPN is a process having message-passing streams among its upper and lower nodes. Each leaf node of CMPN represents a module in CirNet but a non-leaf node represents a block module newly defined in this transformation. Modules specified as a pair by Constr are compiled into an identical node near the leaf of CMPN to assure mutual proximity. The main role of CMPN is the layout generation, i.e., module placement and inter-module wire generation. If the module placement topology, i.e., in what quadrant each subcircuit should be placed, can be given in the stage of CMPN generation, later placement task is fairly simplified.

(SL2) Vertical co-ordination of module shapes

As subproblems are solved in parallel in (S2) above, non-abutment of their final shapes might happen. In that case, chip area will be enlarged due to many dead spaces among modules. To avoid this, the planned shape of the subslice in which a subcircuit is being placed is descended down to the subcircuit as its planned shape. See Figure 2.2(1).

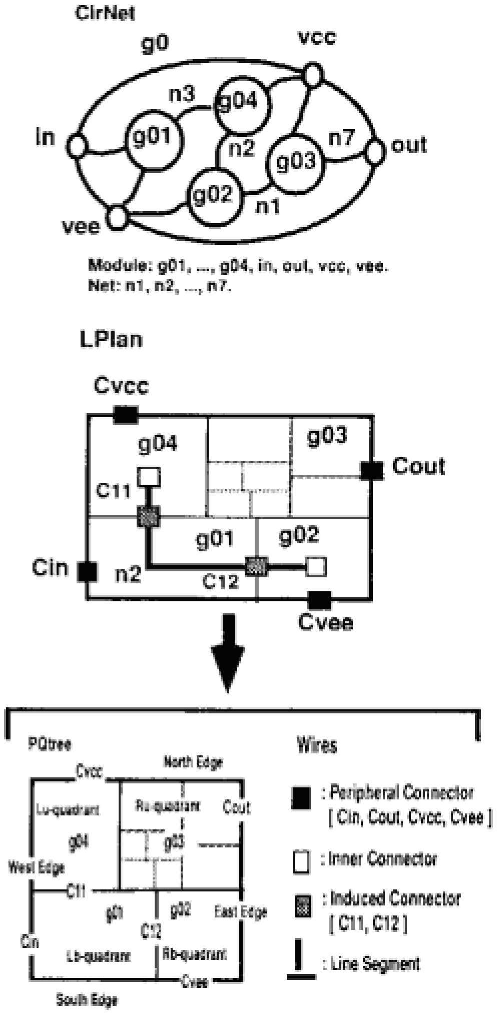


Figure 2.1 Circuit and Layout Representations

(SL3) Horizontal co-operation in wiring

Wire abutment among modules running in parallel had been an unsolved problem in LSI layout design automation. This problem is solved by way of runtime co-operation among nodes in CMPN. The induced connector processes (See 2.1) are used for this purpose. Each of them holds a CERW (Current Existence Range of a Wire on a slice edge). As the first task of wiring, each node process in CMPN tries to narrow CERWs of its peripheral connectors on the north-, west-, south-, and east-edges. If a CERW intersects with two internal subslices, then the CERW is narrowed to one of the two intersections. Among the two candidates, the one which has both enough wiring capacity and minimum wire length with inner connectors is selected. For feed-through nets, i.e., nets having no inner connectors, the CMPN node eagerly waits for a completion of narrowing action by some neighbour node to avoid useless wire bend generation. See

Figure 2.2(2).

(SL4) Wiring direction control to reduce co-operation loads
Although the CERW narrowing co-operation is useful in parallel wiring, it is expensive due to the repeated peripheral connector inspections. This is particulary true for cell wiring where tremendously large number of cells attend in the co-operation. To reduce useless co-operation, wiring direction is co-ordinated in cell wiring. First, in all the cells and for all nets, partial wires are generated that connect inner- and peripheral-connectors on south or east edges. Each CERW on these edges is narrowed into a point where the wire arrived (SE-wiring). Then, NW-wiring follows. Finallly, non-directional feed-through wiring (ND-wiring) is made. Due to the CERW reductions in two previous steps, much eager-waiting co-operation in ND-wiring can be avoided. See Figure 2.2 (3).

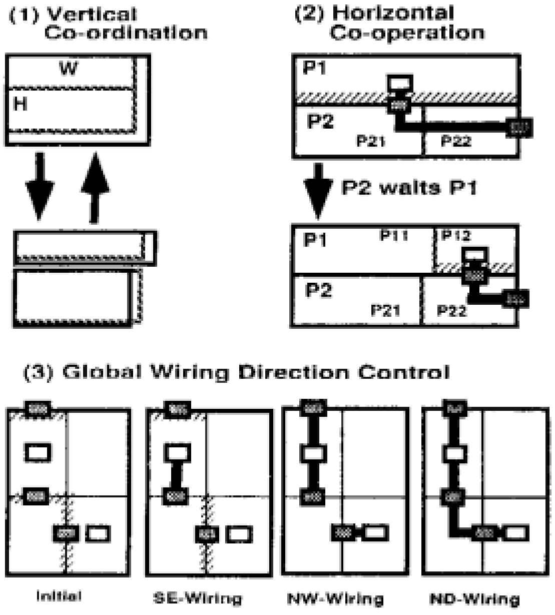

(1) Vertical Co-ordination
W
H
(2) Horizontal Co-operation
P1
P2
P21
P22
P2 waits P1
P1
P11
P12
P2
P21
P22
(3) Global Wiring Direction Control
Initial
SE-Wiring
NW-Wiring
ND-Wiring

Figure 2.2 Problem Solving Heuristics

3 Overview of Co-HLEX

An overview of Co-HLEX is given in Figure 3.1. The main components of Co-HLEX include: a set of original data, I/O functions, a backup memory, a problem solving kernel based on CMPN, and a template library.

3.1 The Problem Solving Kernel

The problem solving kernel is a quadtree-shaped process network CMPN that generates a chip layout. Before the layout generation, each node of CMPN has only circuit data including a module name, the module property, a list of net names connecting this module to others, and a list of subcircuit names. After the layout, a set of layout data is added to each node including; a name of the layoutframe (parameterized slicing template) used to slice the node, an enveloping rectangle size, a list of slicing points in the rectangle, a list of submodule names in each slice, a list of adopted wiring pattern name for each net, and a list of peripheral- and induced-connector names.

3.1.1 Problem Solving Steps

The overall layout problem solving is performed by the follwoing steps.

(ST1) Placement: A placement message containing a PL - a list of planned shape and planned peripheral connector placements - is sent to the top node of CMPN from the top level co-ordination process. Then a set of placement actions based on HRCTL is performed by CMPN processes.

(ST2) Wiring preparation: Upon receiving the placement completion message from the top node of CMPN, the co-ordinator sends a wiring preparation message to it. Then a set of wiring preparation actions are made by CMPN.

(ST3) Wiring non-terminal power nets: Power supply nets - Vcc and Vee - have different width from other signal nets. As they offend the latter, they are wired first. The co-ordinator sends a message to the top node of CMPN to envoke recursive power wiring actions.

(ST4) Wiring non-terminal signal nets: The co-ordinator sends a message to the top node of CMPN to generate signal nets. Then a set of wiring actions based on HRCTL is performed by CMPN processes. Recursion terminates when it reaches to a cell node. At that time, the CERWs held by connector processes contract to the magnitude of cell size.

(ST5) Wiring nets in cells (SE-wiring): The co-ordinator sends a messages to the top node of CMPN to do SE-wiring. This message is passed down to cells. Cells which have inner connectors such as base-, emitter-, collector-contact, etc., that should be wired to peripheral connectors on south or east edges, wires all these nets. After this, each CERW on these edges reduces to a point where a wire reached. As layout rules such as wiring obstacle avoidance, minimun allowances between layout objects, etc., should be sufficed so the maze-router of Lee [Lee 1961] was used with some modifications.

(ST6) Wiring terminal nets in cells (NW-wiring): Similar to that of (ST5).

(ST7) Wiring terminal nets in cells (ND-wiring): The co-ordinator sends a message to the top node of CMPN to do ND-wiring. This is for feed-through wires which only pass above the cell without any inner connectors. All the cell nodes make feed-through wires in parallel co-operation.

3.1.2 Placement by HRCTL

(ST1) Termination of placement: When a terminal cell node in CMPN receives a placement message from above, it imports layout data from relevant layoutframe in the template library.

(ST2) Subproblem generation: When a node - call it CN - in CMPN is a non-terminal module, it generates subproblems as follows. It sends its own PL and a list of its subcircuits with their estimated areas to a subproblem generation planframe (planning frame) in the library.

The planframe sends requests to all layoutframes to make and evaluate possible slicing of PL and embedding of subcircuits into derived slices. The evaluation is made in view of the estimated wire length among inner- and peripheral-connectors, the estimated layout area, and estimated distortions of realized layout from the PL. Receiving the best plan and the relevant layoutframe name from the planframe, CN memorizes them. Then inter-slice wiring is made by a wiring planframe relevant to the chosen layoutframe. In the first step of this wiring, CERWs of peripheral connectors are narrowed. Then inter-slice wires are planned for each net. Only abstract wiring plan is made as shown in Figure 2.1. Wiring patterns attatched to the chosen layoutframe is used. As the final wireability largely depends on the wire congestion on slice edges, so the wiring resource consumption on these edges should be balanced. To do this, the idea of wiring resource vector is introduced. It is a list of maximum possible wires through slice edges. In selecting a wiring pattern for each net, the resource vector consumption is analyzed for all the possible patterns and the best one is selected. New induced connectors are given, each having a CERW identical to the edge length on which it was defined. Induced connector processes are newly spawned, each having a message stream to CN. Finally, subproblem definition planframe is envoked to give all the PLs for all subcircuits. The planframe defines PLs by using derived subslices and their peripheral connectors and descends them to lower CMPN nodes. Streams to peripheral connectors are also descended to assure subsequent narrowing actions by subnodes. Notice that by this combination of placement and wiring, PLs of subcircuits homologous to that of parent CN could be generated.

(ST3) Recursion: All the subnodes of CN are invoked in parallel to solve their problems. When many PEs are available, they are spawned on different PEs.

(ST4) Placement aggregation: Layout aggregation planframe is invoked by CN. It waits for the completion message from all subnodes and after eceiving the message, it aggregates all the realized layouts of subnodes to generate the CN layout. The layoutframe chosen in (ST2) is reused here to give an aggregation scheme. But when it gives a large dead space in CN layout, layoutframe swapping is tried.

3.1.3 Wiring Preparation

To make dead spaces usable in wiring, they are compiled into CMPN as dummy modules. Module placement points are determined in world co-ordinate with its origin at noth-west corner of the chip. After the dead space compilation, connector processes generated in placement become unusable, so they are killed. A CWPN - Cell Wiring Process Network approximating the meshed cell - is newly generated under each cell sharing a communication stream with the cell. Wiring obstacles in each cell are examined by using the relevant layoutframe and written into CWPN.

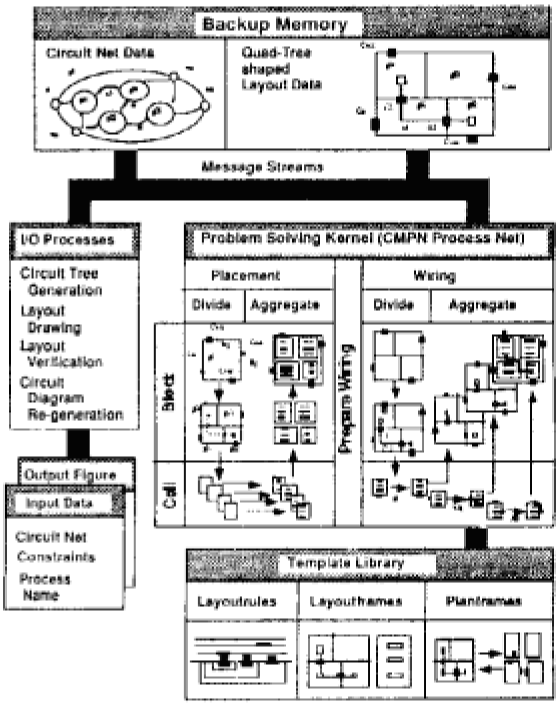


Figure 3.1 Overview of Co-HLEX

3.1.4 Non-terminal Power Net Wiring by HRCTL

(ST1) Termination of wiring: When CN is a terminal node having no inner power connectors, it only ascends a completion message to its upper node. Otherwise, a power wire termination planframe is invoked to generate power-wire connectors in the cell.

(ST2) Subproblem generation: When CN is a non-terminal block with inner power connectors, a planframe is invoked to extend power wires along slice edges reaching to subslices.

(ST3) Recursion: All the subnodes of CN run in parallel to make power wires.

(ST4) Wiring aggregation: Aggregation planframe for power wire is invoked by CN. It waits for the power wiring completion messages from subnodes and determines the width of CN power lines based on electrical considerations.

3.1.5 Non-terminal Signal Net Wiring by HRCTL

(ST1) Termination of wiring: When CN is a terminal node, it only ascends a completion message.

(ST2) Subproblem generation: When CN is a non-terminal block, it generates wiring subproblems by using the same method as explained in 3.1.2 (ST2). As the module placement is already given, only wiring action is repeated. CERWs of peripheral connectors are narrowed first. Then wires are made giving new induced connector processes. Their names are descended down to subnodes for recursion.

(ST3) Recursion: All the subnodes of CN run in parallel to solve their problems. When many PEs are available, they are spawned on different PEs.

(ST4) Wiring aggregation: When CN receives completion messages from all subnodes, it ascends its own completion message.

3.1.6 Wiring Terminal Nets

(ST1) Wirable net detection: Each terminal CN, in this case a terminal cell, finds a net that has at least one fixed peripheral- or inner- connector. If such a net is found, CN broadcasts the net name, its connectors, and usable wiring layers at these connectors to its CWPN.

(ST2) Pre-processing: Using the broadcasted information, CWPN changes the passage cost on its nodes. High costs are given to nodes in wire inhibition area.

(ST3) Wiring a net: Modified maze-routing is performed on CWPN to find wires among given connectors.

(ST4) Post-processing: Upon completion, the CWPN node on which a connector or a wire is placed is given a high passage cost. Finally a completion message is sent to CN. Then other net is tried from (ST1).

(ST5) Wiring other nets: After memolizing the reported wire, the cell repeats step (ST1) untill all nets are wired.

3.2 Template Library

3.2.1 Plan Frames

Planframes are a set of procedures used by CN as explained in 3.1. Many of them are layoutframe specific.

(1) Choose an appropriate layoutframe for subproblem generation.

(2) Evaluate a proposed plan for placement or wiring.

(3) Generate subproblems for placement or wiring.

(4) Descend subproblems down to subnodes.

(5) Aggregate subproblem solutions for placement or wiring.

(6) Other functions.

3.2.2 Layout Frames

Layoutframes are templates, or types in other words, for layout.

(1) Block level layoutframes: Slicing templates of arity 1, 2, 3, and 4 containing several slicing structure variants. Template specific wiring patterns are included.

(2) Cell level layoutframes: Parameterized configuration templates of transistors, resisters, capacitors, and connectors.

3.2.3 Layout Rules

(1) Cell size definitions (depends on fabrication process).

(2) Allowances (same, admissible gaps between objects).

(3) Wiring rules (same, wiring layers and their usabiliy by signal and power nets).

3.3 I/O Functions

3.3.1 CMPN Generator

(ST1) Input data: The original circuit net CirNet and the PL of the topmost chip.

(ST2) Process network generation: Circuit modules and nets are transformed into processes and their connections are replaced by streams giving a CMPN - Circuit Module Process Network.

(ST3) Module shape alignment: Align-shape message is broadcasted to CMPN from the top co-ordinator. Divisible modules in CMPN such as resisters divide themselves to give aligned heights to those of standard transistors. As a result, an enlarged CMPN is given.

(ST4) Hierarchy generation: The flat CMPN given by (ST3) is recursively partitioned to give a hierarchical CMPN.

3.3.2 Assignment of Processes on PEs

LAP (List of available processors) is given to the top node of CMPN. The top node divides LAP into the number of its subnodes in accordance with their computation loads. As an approximation of the load, total number of modules in the circuit is used. One of the PE is picked up from each subset and a subnode is spawned on the PE. This process is recurred until LAP becomes indivisible. After that, all subnodes in CMPN is spawned on the same PE.

3.3.3 Other Functions

Layout data in CMPN is written out to a display terminal.

4 Computational Complexity of HRCTL

Definitions.
Let PrBPT(R,N) denotes a balanced CMPN with R subnodes and height N. Let leaf(PrBPT(R,N)) denotes the number of leaf nodes of PrBPT(R,N). Let no(PEs) denotes the number of parallel processing elements on which the problem PrBPT(R,N) is solved by the HRCTL algorithm.

Suppositions.
All the nodes of PrBPT(R,N) consume the same computation power. Instantaneous communication among processes is possible without any computation load. The total elapsed time of processing on one PE is proportional to its total computation load.

Theorem.

For PrBPT, HRCTL has the time complexity of either
O(log(leaf(PrBPT(R,N))))
or O(log(no(PE))+leaf(PrBPT(R,N))/no(PE)).
The latter is the usual case where large problem is solved on limited PEs.

Proof.

Case1. no(PE) >= leaf(PrBPT(R,N)) : The president PE is the bottleneck processor which receives the topmost node of PrBPT(R,N). It processes maximum number of nodes among PEs. The maximum number is log(leaf(PrBPT(R,N))).
Case2. no(PE) =< leaf(PrBPT(R,N)) : The president PE is also the bottleneck processor. Until the depth of log(no(PE)) is reached on PrBPT(R,N), case1 applies. After it, each PE is obliged to solve all the unsolved nodes in pseudo parallel mode. Here, the number of unsolved nodes is leaf(PrBPT(R,N))/no(PE). As the president PE faces the two situations sequentially, they should be added to give the log(no(PE))+leaf(PrBPT(R,N))/no(PE) complexity. **QED.**

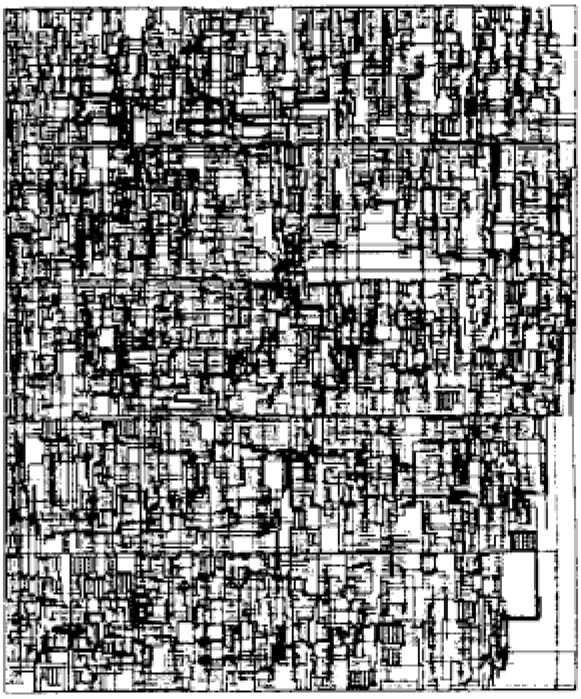
5 Experiments

5.1 Experiment Design

(1) Main objectives: The main objectives of the experiment are the verifications of:
OE1. Parallel placement and wiring capability,
OE2. Wire length and chip area reduction by vertical co-ordination and holizontal co-operation,
OE3. Enhanced computation speed,
OE4. The program size reduction and maintenability.
(2) Used circuit and fabrication process: A real bipolar analog circuit with 1019 modules and 683 nets are used in the experiment. 149 pairs were given as Constr. After module shape alignment, CMPN had 1299 modules and 901 nets. The height of generated CMPN was 14. From this original circuit, 254-, 489-, and 810-moduled subcircuits were extracted for computation speed measurement. A bipolar analog fabrication process with 3 wiring layers of AL1, AL2, and AL3 was assumed. The first two are for signal nets and the last one for power nets. For signal nets, as much AL1 should be used as possible to attain high electrical quality. Traditionally, time consuming maze-router is usually applied to this problem.

5.2 Experiment Results

Figure 5.1: Example chip layout.
Figure 5.2: Computation speed.
Figure 5.3: PEs vs Speedup.
Table 5.1 : Scale of Co-HLEX.



1299 modules, 683 nets, 623 sec/Multi-PSI.64PEs
Figure 5.1 Bipolar Analog Circuit Layout by Co-HLEX

5.3 Considerations

(1) The possibility of parallel layout problem solving.
This has been proved through the experiment. As far as we know, Co-HLEX is the first system that can abut layouts - module shapes and wires - by runtime co-operation.
(2) Quality of Generated Layout: Wire length and chip area.
Through observations of Figure 5.1 we notice that both compact module placement and wires without useless bends could be generated. By the runtime wire abutment co-operation, traditional channel areas to patch inter-submodule wires could be diminished. This contributes to chip area reduction.
(3) Computational efficiency.
Figure 5.2 shows the performance of both Co-HLEX on Multi-PSI/64PE and a practical layout system on a main frame. Co-HLEX has a time complexity of nearly $O(N^{1.0})$. Here N is the number of modules in the circuit. For the 1299 moduled circuit, it took only 623 sec. This extraordinary outperforms the traditional system. Also, nearly linear peedup could be attained as shown in Figure 5.3.
(4) Program size and maintainability.
The 6,000-lined Co-HLEX remarkably outperforms the $O(10^5)$~$O(10^6)$-sized traditional implementations. See

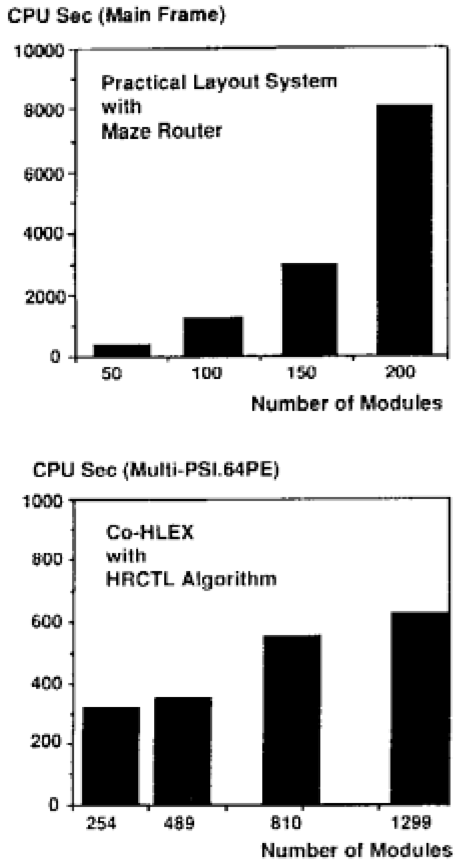


Figure 5.2 Problem Size vs Problem Solving Time

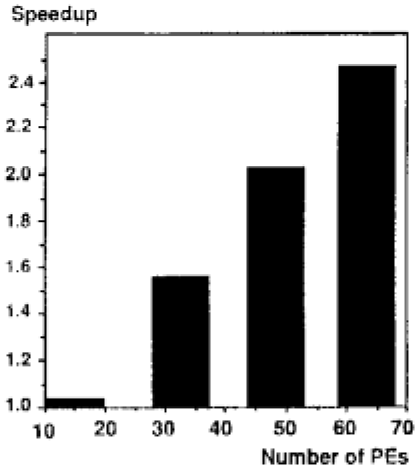


Figure 5.3 PEs vs Speedup

Table 5.1 Scale of Co-HLEX

Subsystems	KL1.Lines
Kernel	620
Planframes	2648
Layoutframes	1180
Layoutrules	684
Utilities	865
Total	5997

Table.5.1. This is due to the recursive HRCTL algorithm. Highly modularized program description was possible on streamed-parallel dataflow computation model offered by KL1.

6 Conclusions

6.1 Results

(1) A co-operative hierarchical layout problem solver named Co-HLEX was developed in FGCS project as an application program of Fifth Generation Parallel Inference Machines.
(2) The kernel algorithm of Co-HLEX is HRCTL which is a hierarchical recursive concurrent theorem prover for layout. Taditional wiring channels could be avoided due to its runtime co-operation to abut module shapes and wiring connectors.
(3) Due to the recursive nature of HRCTL, Co-HLEX is nearly 6,000-lined in KL1 which remarkably outperforms traditional LSI layout program implementations.
(4) Nearly O(N$^{1.0}$) time performance could be attained due to the streamed-parallel and distributed-memory architecture of Co-HLEX which greatly outperforms traditional methods.

6.2 New Problems

Programmers are in the well-known plan-do-see cycle. The non-repeatablity of parallel computation often destroyes the loop and deteriorates debugging efficiency. A programming environment for parallelism would be one of the most important issues to be studied further.

Acknowledgements

We acknowledge Dr. K.Furukawa, Dr. K.Nitta, Dr. K.Taki and other ICOT members for general supports, Prof. M.Iri of the University of Tokyo for advices on the problem solver design, Mr. M.Oda, Mr. H.Yamaguchi, Mr. K.Kawamura, Mr. Y.Mori, and Y. Itoh of ISC Ltd. for assisting the program development, Mr. S.Domen and Dr.K.Ishihara; the general manager and the senior chief researcher of the Systems Development Lab. of Hitachi, for their supports of the research, Mr. K. Maio of the Semiconductor Design and Development Center, Hitachi Ltd., for assisting layout experiments, Dr. Y.Shiraishi of the Central Research Lab. of Hitachi for state of the art discussions, and Dr. S.Hayashi of Systems Development Lab. of Hitachi for discussions on analog circuit layout.

References

[Kleene 1952] S. C. Kleene. Introduction to Metamathematics, Van Nostrand,1952.

[Mandelbrot 1982] B. Mandelbrot. The Fractal Geometry of Nature, W. H. Freeman, 1982.

[Watanabe and Hayashi 1989] T. Watanabe, and S.Hayashi. Modelling Layout Problem Solving in Logic, in CAD Systems using AI Techniques, G. Odawara(ed.), pp.181-188, North-Holland, 1989.

[Samet 1984] H. Samet. The Quadtree and Related Hierarchical Data Structures, Computing Survey, Vol.16, No.2, June (1984), pp.187-260.

[Otten 1982] R. Otten. Automatic Floorplan Design, Proc. 19th DAC (1982), pp.261-267.

[Luk et al. 1986] W. K. Luk, D. T. Tang, and C. K. Wong. Hierarchical Global Wiring for Custom Chip Design, Proc. 23rd DAC (1986), pp.481-489.

[Lee 1961] C. Y. Lee. An Algorithm for Path Connections and its Applications, IRE TEC, September (1961), pp.346-365.

[Watanabe and Komatsu 1991] T. Watanabe and K. Komatsu. Co-operative Hierarchical Layout Problem Solver on Parallel Inference Machine, Proc. of the LPC'91, ICOT (1991) pp.9-24.