TR-707

Co-HLEX: LSI Layout System on Japan's Fifth Generation Parallel Inference Machine

by
T. Watanabe & K. Komatsu
(Hitachi)

October, 1991

Co-HLEX: LSI Layout System 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

Co-HLEX is a cooperative 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 through message passing cooperations. 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 a 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 layout of 1 cm^2 DRAM chip can include so much rectangles that can pave roads of 10 m width all over the Europe. New ideas including parallel computation are greatly required to cope with this complexity.

(R2) The development and enhancement of a layout system, with more than 1 million-lined program codes, consume huge amount of programmers' unrewarded labour. One of our ideas is the use of recursion principle to reduce it[1,2]. The other is the use of streamed parallel dataflow computation model to represent mutually related objects in well modulalized message driven processes. We nicknamed our algorithm as HRCTL as an abbreviation of "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 it.

(R3) The classical divide and conquer - the hierarchical problem solving - works well as long as subproblems correlates weakly. In case of LSI layout, this premise cannot be sufficed. Neighbouring modules should have abutted shapes and wires to avoid dead spaces. Our idea is the use of communications among modules to solve this AND-typed constraint. In the following, Section 2, 3, 4, and 5 gives basic concepts, system overview, a brief complexity consideration, and experiment results, respectively. Based on these results, 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 the 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 representing placement by a slice hierarchy[4,5,6], 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 terminal points, vias, and induced connectors of a net.
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 a line spanning two connectors for a net. Connector names on both sides, run layer, and line width are also included.

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

Constr:= List of proximity conditions for modules, usually called "Pairs". Topmost planned layout - a planned chip size (Width and Height) and planned peripheral connector placements - is included.

2.2. Recursive Layout Problem Solving

The slicing structure representation permits us the following elegant and simple recursive problem solving.

(S1) If the module is an indivisible leaf cell, import it's layout from a library. If the module is a divisible block, divide the original problem with a circuit data and a 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 elemets or PEs are available, fork them on different PEs.

(S3) Aggregate finished layouts of subproblems following the division policy given by (S1) to generate the layout of the original problem.

2.3. Problems and Solutions

(S11) 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 sub-circuits, and their embedding plan into slices should be made. To avoid the untractable computational complexity caused by these two divisions and one embedding, we translate the CirNet 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 it's upper and lower nodes. A leaf node of CMPN corresponds to a module in CirNet but a non-leaf node is a block module newly defined in this translation. Modules specified as pairs by Constr are compiled into an identical node near the leaf of CMPN to assure mutual proximity. Then CMPN is used to generate the layout, i.e., module placements and inter-module wires. If module placement topology, i.e., in what quadrant each sub-circuit should be placed, can be given in the stage of CMPN generation, the later placement task becomes a simple recursive module shape determination.

(Sl2) Vertical coordination of module shapes

As subproblems are solved in parallel in (S2) above, they might realize non-abutted final shapes each other. In that case, we have a large-areared chip with many dead spaces among modules. To avoid this, the planned shape of the slice in which a sub-circuit is embed is descended to the sub-circuit as its planned-shape. See Figure 2.2(1).

(Sl3) Horizontal cooperation in wiring

Wire abutment among modules running in parallel had been an unsolved problem in LSI layout design automation. We solve this problem by way of runtime cooperations among nodes in CMPN. We introduce induced connector processes defined in 2.1. It holds the current existence range - CERW - of itself on the slice edge. After a node process in CMPN completes the sub-circuits' embed into it's slices in (S1), the node process tries to narrow CERWs of it's peripheral connectors on it's north-, west-, south-, and east-edges. If a CERW is wide enough to overlap with two slice borders, then the CERW is narrowed so as to overlap with only one border. Among the two narrowing candidates, the one which has the minimum wire length with inner sub-circuits is selected. When the narrowing actions are made by face-to-face modules in concurrent mode, the module which has no inner connetor to connect with the peripheral connector waits the narrowing action of the other to avoid useless wire bendings. See Figure 2.2(2).

(Sl4) Global wiring direction control for useless cooperatiŏn reduction

Although the message-passing cooperation for CERW narrowing is a key in parallel wiring, it is expensive due to it's repeated peripheral connector range inspections. This is particulary true for cell wiring where tremendously large number of cells attend in the cooperation. To reduce useless cooperations, wiring direction is coordinated in cell wiring. First, wires are made into north-west direction in all the cells starting from inner connectors, reducing the CERWs on north or east edge into a point (NW-wiring). Second, SE-wiring. Finallly, non-directional feed-through wiring(ND-wiring) is made by cooperations. See Figure 2.2 (3).

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 sub-circuit names. After the layout, the layout data including a layoutframe name used to slice it, an enveloping rectangle size, a list of slicing points in the rectangle, sub-module names in each slice, a list of adopted wiring pattern names for each net, and a list of peripheral- and induced-connector names are added to each node.

3.1.1. Problem solving steps

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

(St1) Placement: A placement message containing a planned layout - a list of planned shape and planned peripheral connector placements - is sent to the top node of CMPN from the top level coordination 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 coordinator 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 at first. The coordinator sends a message to the top node of CMPN to do power wiring. Power wires are paved using a simple recursive algorithm until it reaches each cell node in the CMPN.

(St4) Wiring non-terminal signal nets: The coordinator 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 at each cell node, so the CERWs held by connector processes contract to the magnitude of cell size but not to a point.

(St5) Wiring nets in cells (NW-wiring): The coordinator sends a messages to the top node of CMPN to generate cell wires towards north-east-directions. 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 north or west edges, wires all these nets. After this, each CERW on these edges is contracted into a point where a wiring path reached. Layout rules such as wiring obstacle avoidance, minimun distance assurance, etc., should be sufficed here. We use the maze-router of Lee[7] with some modifications.

(St6) Wiring terminal nets in cells (SE-wiring): Similar to that of (St5).

(St7) Wiring terminal nets in cells (ND-wiring): The coordinator sends a message to the top node of CMPN to generate cell wires by non-directional-wiring. This is for feedthrough wires which only pass through the cell without any inner connectors. All the cell nodes make feed through wires in parallel cooperation.

3.1.2. Placement by HRCTL

(St1) Termination of placement: When a terminal node in CMPN receives a placement message from above, it only returns back a completion message. In case of a cell node, 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 it's planned-layout and a list of sub-circuits with estimated areas to a subproblem generation planframe in the library. The planframe sends requests to all layoutframes to make and evaluate possible slicing of the planned-layout and embedding plans of sub-circuits into slices. Here the layoutframe denotes a slicing template in the template library. The evaluation is made in view of the estimated wire length among sub-circuits and edge connectors, the estimated layout area, and estimated distortions of realized layout from the planned layout. The planframe reports the best slicing plan and the relevant layoutframe name to CN. Then inter-slice wiring is made by CN. In the first step of this wiring, CERWs of peripheral connectors are narrowed. Then inter-slice wires are planned. The wiring made here are only abstract ones as shown in Figure 2.1. Wiring patterns attatched to the chosen layoutframe is used to generate wires. As the final wireability largely depends on the wire congestion on slice edges, so the wiring resourse 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 pattern is chosen. We can get new induced connectors each having a CERW identical to the edge length on which the connector resides. Processes for induced connectors are newly spawned, each with a message stream to CN. Finally, the derived slice shapes and connectors on their border edges are gathered to define planned-layouts of all sub-circuits. The CN informs them to lower nodes as their planned-layouts. Streams to connectors are also descended to assure subsequent narrowing actions in lower nodes. Notice that by this combination of placement and wiring, planned-layouts of sub-circuits which are homologous to that of CN could be generated.

(St3) Recursion: All the lower nodes of CN run in parallel to solve their problems. When many PEs are available, they are spawned on different PEs.

(St4) Placement aggregation: When CN receives completion messages from all subnodes, it aggregates all their realized slices to form it's shape. The aggregation is done by using several layoutframes including the one selected in (St2) to generate most compacted layout.

3.1.3. Wiring preparation

Dead spaces generated in placement is compiled into CMPN as dummy modules. By this, deadspaces become to be usable in the subsequent wiring. Module places are precisely determimed in the world coordinate with it's origin at noth-west corner of the chip. Due to the CMPN deformation, connector processes generated in placement become useless, so they are deleted. A CWPN - Cell Wiring Process Network - is generated under each cell with a communication stream to the cell. Also, wiring obstacles are read in from relevant layouframe of each cell and memolized in the CWPN.

3.1.4. Non-terminal power net wiring by HRCTL

(St1) Termination of wiring: When CN having no power nets receives a wiring message, it only returns back a completion message to it's upper node. When CN has power nets, power-wire connectors for the cell are generated by using relevant layoutframe.

(St2) Subproblem generation: When CN is a non-terminal block, it extends power wires along slice edges to those slices that have inner power net connectors.

(St3) Recursion: All lower nodes of CN run in parallel to make power wires.

(St4) Wiring aggregation: When CN receives power wiring completion messages from all subnodes, it determines the final width of it's power lines and asends a completion message.

3.1.5. Non-terminal signal net wiring by HRCTL

(St1) Termination of wiring: When CN is a terminal node or a cell node, it only reports 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 given already, only wiring action is repeated. New connector processes are introduced as before and descended below.

(St3) Recursion: All the subnodes 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 a completion message.

3.1.6. Wiring terminal nets

(St1) Wiring a net: The cell picks up a net that becomes to have at least one fixed peripheral connector and sends a list of connectors of the net to CWPN. Modified maze-routing algorithm is used to find wires among them. Upon completion, wires and connectors for the net are added in CWPN as new obstacles. Results are reported to the cell.

(St2) 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. Planframes

Planframes are following predicates used by CMPN nodes in their tasks explained in 3.1.

(1) Choose an appropriate layoutframe for subproblem generation.

(2) Evaluate proposed plan (for placement and wiring).

(3) Generate subproblems (same).

(4) Aggregate subproblem solutions (same).

(5) Other functions.

3.2.2. Layoutframes

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

(1) Block level layoutframes: Slicing patterns of arity 1, 2, 3, and 4 with several sub-patterns with different slicing structures. Wiring patterns are included in each of them.

(2) Cell level layoutframe: Configuration definitions for transistors, resisters, capacitors, and vias.

3.2.3. Layout rules

(1) Fabrication process dependent wiring layers and their usage by signal and power nets.

(2) Fabrication process dependent permissible minimum gap among objects on each layer.

(3) Fabrication process dependent.

3.3. I/O Functions

3.3.1. CMPN generator

(St1) Input data: The original circuit net CirNet and the planned layout of the chip.

(St2) Process network generation: Circuit modules and nets are transformed into processes and their connections are replaced by streams. We get a CMPN - Circuit Module Process Network.

(St3) Module shape alignment: Align-shape message is broadcasted to CMPN from the top coordinator. Resister and capacitor processes in CMPN divide themselves to give nearly abutted heights with standard transistors. As a result, CMPN is enlarged.

(St4) Hierarchy generation: The flat CMPN given by (St3) is recursively partitioned to give a hierarchical CMPN. Details are skipped in this paper.

3.3.2. Assignment of processes on PEs

List of available processors LAPE is given to the top node of CMPN. The top node divides LAPE into the number of subnodes in accordance with their computation loads. We use the total number of modules in a circuit as it's computation load approximation. One of the PE is picked up from each subset and a subnode is spawned on the PE. This process is recursed until LAPE becomes indivisible. After that, all the nodes in CMPN is spawned on the same PE.

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 coordination and holizontal cooperation,
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 becomes to have 1621 modules and 1223 nets. The height of generated CMPN was 14. Duplication and deletion of this original sub-circuit is done to give 296-, 573-, and 1593-moduled circuits for computation speed measurement. We assumed a bipolar analog fabrication process with 3 wiring layers of AL1, AL2, and AL3. The first two are mainly for signal nets and the last for power nets.

5.2. Experiment Results

Figure 5.1: The resulted chip layout.
Figure 5.2: Computation speed.
Table 5.1 : Scale of co-HLEX.

5.3. Considerations

(1) The possibility of parallel layout problem solving.
This has been completely proved through our experiment. As far as we know, co-HLEX is the first system that can abut module layouts - shapes and wires - by runtime cooperations.
(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 cooperaions, channel areas wherein inter-module wires are patched after parallel layout of them could be diminished. This greatly reduces the total chip area in comparison with traditional methods.
(3) Computation time vs problem size.
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 1019 moduled circuit, it took only 60 sec. This extraordinary outperforms the traditional system.
(4) Program size and maintenability.

The 6,000 lined co-HLEX remarkably outperforms the $O(10^5)$-$O(10^6)$ sized traditional implementations. See 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 cooperative 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. Due to the concurrent cooperations, module shape and wire connector abutment become possible avoiding traditional wiring channels.

(3) Due to the recursive nature of HRCTL, co-HLEX is nearly 6,000 lines in KL1 which remarkably outperforms traditional LSI layout program implementations.

(4) The streamed-parallel and distributed-memory architecture attained nearly $O(N^{1.0})$ performance on 64PEs, 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. 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, and Mr. M.Oda, Mr. H.Yamaguchi, Mr. K.Kawamura, Mr. Y.Mori, and Y. Itoh of ISC Ltd. for 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 the experiment, 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.

REFERENCES

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

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

[3] Watanabe, T. and Shinichi, H., Modelling Layout Problem Solving in Logic, in CAD Systems using AI Techniques, G. Odawara(ed.), pp181-188, North-Holland.

[4] Samet, H., The Quadtree and Related Hierarchical Data Structures, Computing Survey, Vol.16, No.2, pp187-260, June, 1984.

[5] Otten, R, Automatic Floorplan Design, Proc. 19th DAC, pp261-267, 1982.

[6] Luk, W.K., Tang, D.T., and Wong, C.K., Hierarchical Global Wiring for Custom Chip Design, Proc. 23rd DAC, pp481-489, 1986.

[7] Lee, C.Y., An algorithm for path connections and its applications, IRE TEC, pp.346-365, September, 1961.

[8] Watanabe, T. and Keiko, K., Co-operative Hierarchical Layout Problem Solver on Parallel Inference Machine, Proc. of the LPC'91, pp9-24, ICOT, 1991.

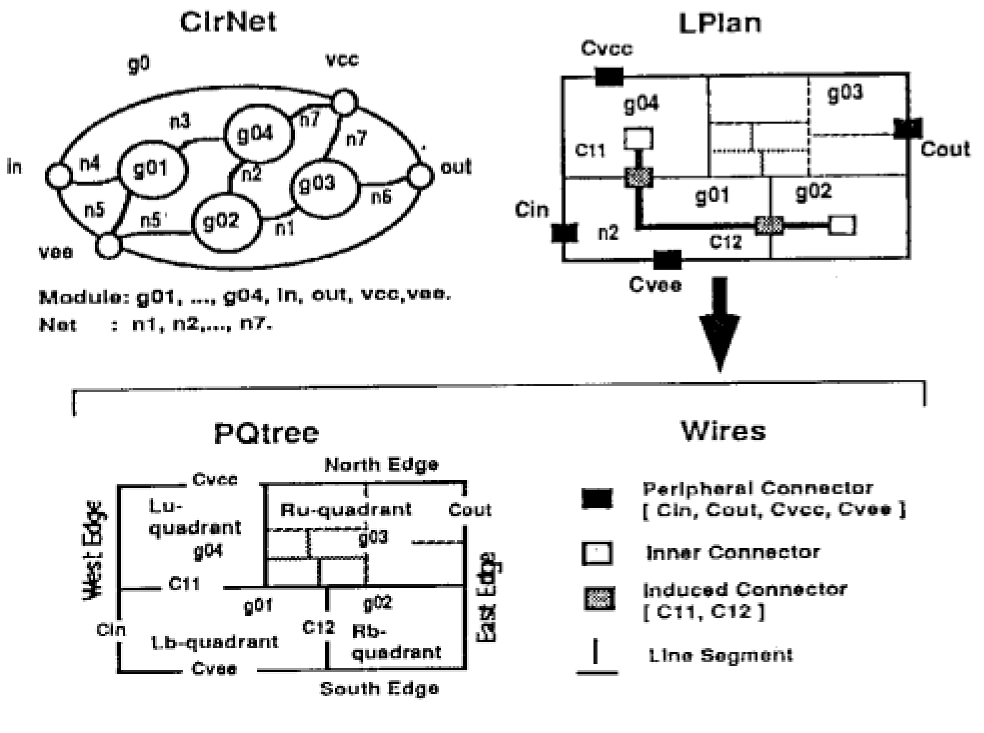
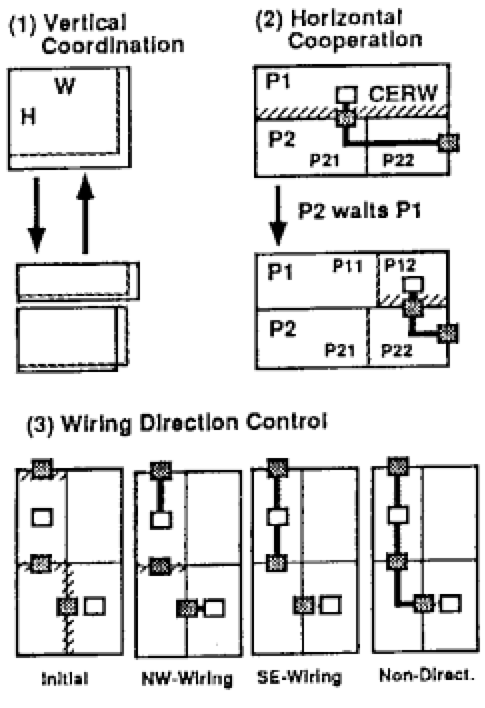


Figure 2.1 Ciruit and Layout Representation



(1) Vertical Coordination
W
H
(2) Horizontal Cooperation
P1
CERW
P2
P21
P22
P2 waits P1
P11
P12
(3) Wiring Direction Control
Initial
NW-Wiring
SE-Wiring
Non-Direct.

Figure 2.2 Problem Solving Heuristics

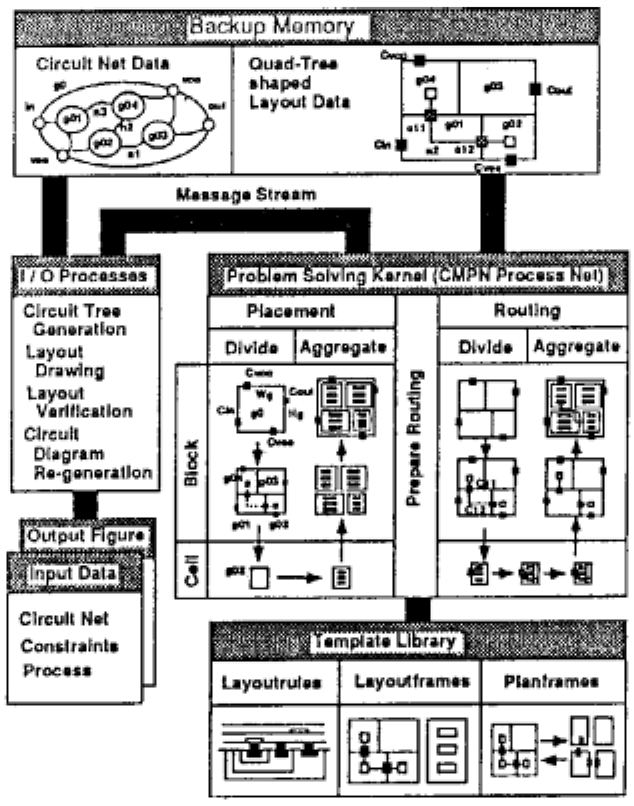
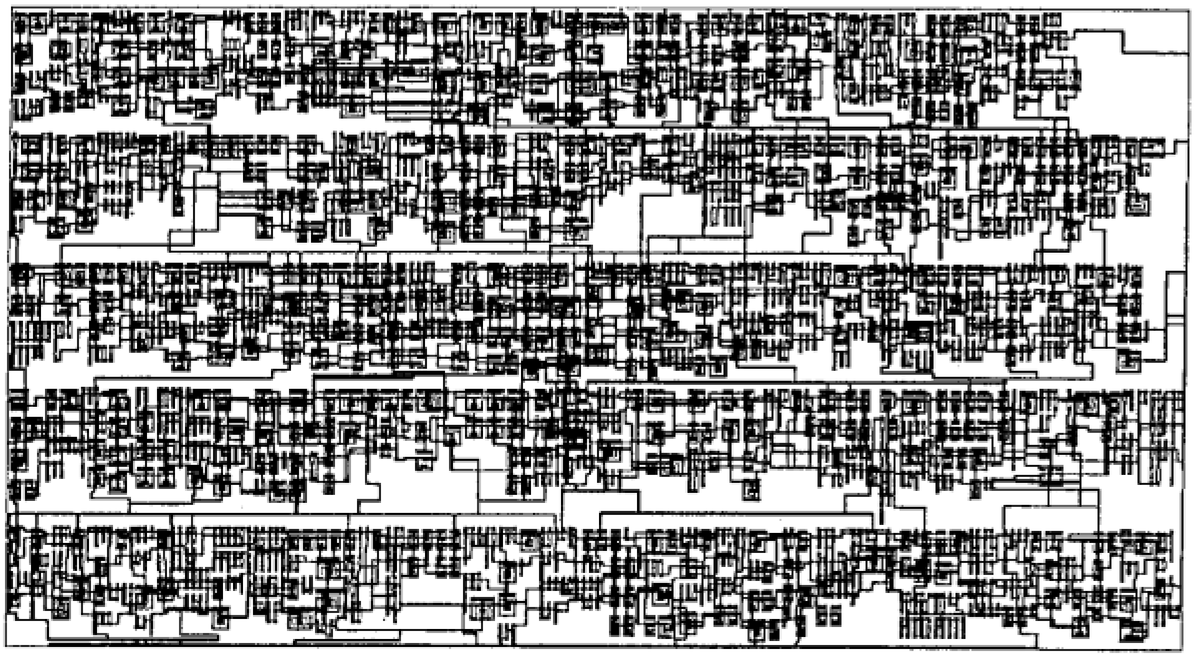


Figure 3.1 Overview of co-HLEX



1019 Modules, 683 Nets, 59.5 sec / Multi-PSI.64PE

Figure 5.1 Bipolar Analog Circuit Layout by co-HLEX

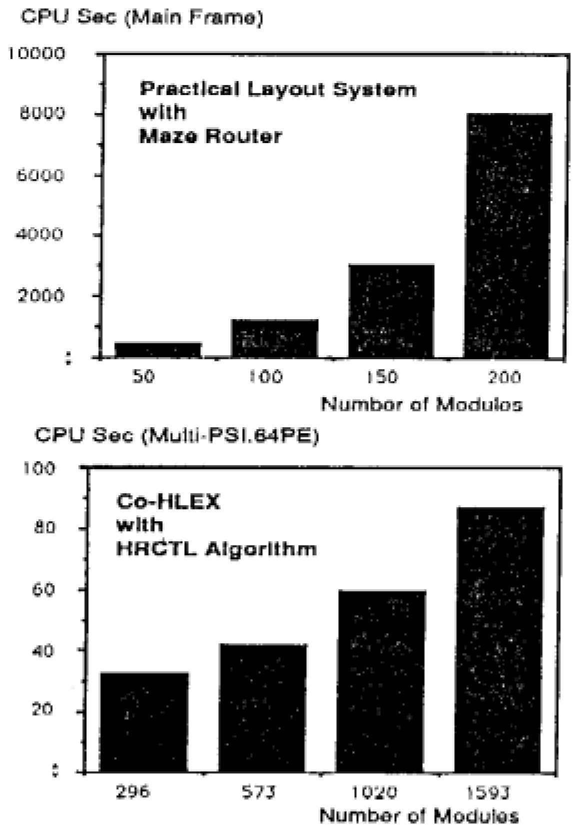


Figure 5.2 Problem Size vs Problem Solving Time

Table 5.1 Scale of Co-HLEX

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