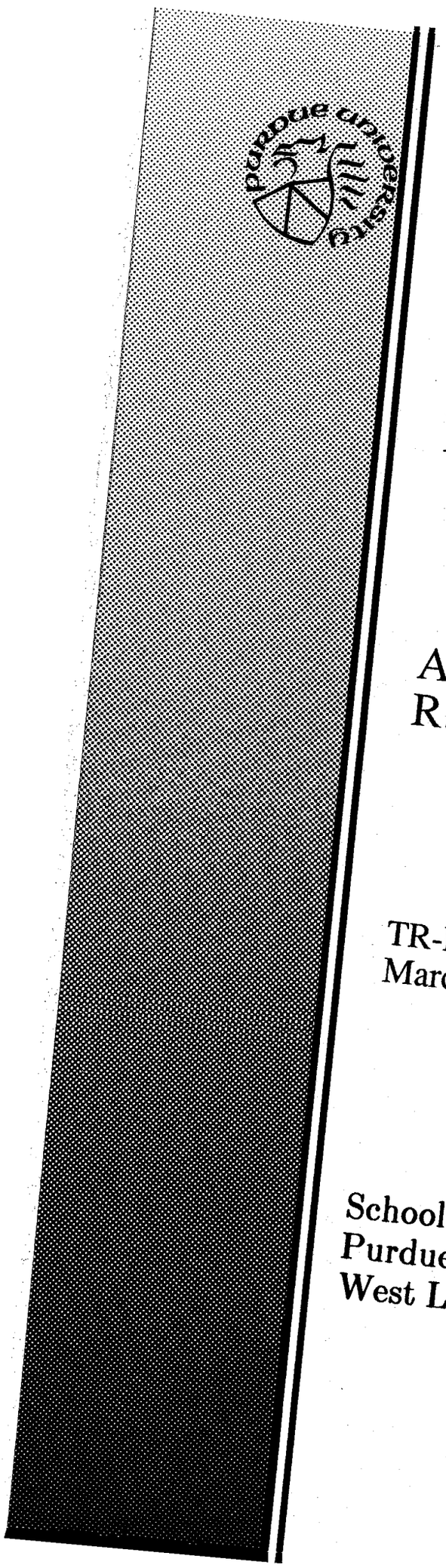
3-1-1990

# A Circuit Layout Methodology Incorporating Machine Learning

A. Ford
*Purdue University*

R. Fujii
*Purdue University*



# A Circuit Layout Methodology Incorporating Machine Learning

A. Ford
R. Fujii

TR-EE 90-20
March 1990

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

# A Circuit Layout Methodology Incorporating Machine Learning


## Abstract

We describe the use of a form of machine learning which makes efficient use of time and computing resources for developing and optimizing a transistor level IC layout. Our learning system abstracts new knowledge from examples it is provided and incorporates this new knowledge into a generalized solution graph. This generalized solution graph contains information about layout topologies known to the system.


## I. Introduction

The process of determining a solution to a given circuit layout problem involves: a) acquiring knowledge from user-provided examples; b) partitioning the problem into sub-circuits for which layout solutions are already known; c) interacting with the system user in those areas of the problem for which layout solutions are not known; d) interconnecting the sub-layouts for which layout solutions are known to obtain a problem solution graph; and e) mapping the generalized problem solution graph into a specific technology or application.

To learn, examples are generalized and combined to form a single "Generalized Solution Graph" which contains the known topologies for laying out circuits. This generalization allows a the graph to be used for circuits which share the same schematic but differ in external connections.

To partition a problem circuit, the generalized solution graph is compared to the problem circuit to find a largest common sub-circuit; this becomes the first partition and its layout is already known. The remainder of the problem circuit is a smaller layout problem and is solved by recursively subdividing in a similar manner, until either all of the problem circuit's elements have been assigned to a partition or until only elements that cannot be assigned to a partition remain. The only elements which will not be assigned to a partition are those that the system has not been taught in the examples and the user must be consulted for examples containing these elements if the system is to complete the partition.

If the result of the partitioning is only one partition, then a layout for the entire circuit is known and is called the "Problem Solution Graph." In the case of a single partition, elements in the generalized solution graph which are not in the problem circuit are deleted, leaving the problem solution graph. Otherwise, the multiple partitions represent the pieces that will be connected to derive a problem solution graph. As in the case of a single partition, the elements in the generalized solution graph which are not in a partition are deleted, leaving a partition layout

graph for that partition. A layout graph for each partition is determined and these pieces are then placed and interconnected. One way to accomplish this is to place and connect the pair of partitions that have the greatest number of interconnections between them. This process reduces the number of partitions by one and can be recursively applied until only one partition remains. The graph for the single partition is a derived problem solution graph.

The effort required by the proposed system to solve a particular problem is directly related to the "similarity" of the problem to the user-provided examples and problems it has solved successfully. Considerable similarity implies solutions involving minor alterations of the user's examples can be rapidly found. Little similarity causes layouts of relatively small subsets of the known circuits to form the units that must be positioned and interconnected.

This methodology has the potential to become the "next generation" circuit-level standard cell implementation technique. Current standard cell techniques rely on a limited library of gate-level cell libraries to implement a design. The proposed methodology learns to lay out circuits from past examples and it adds new knowledge to the database whenever a problem which has not been solved previously is encountered; thus the database of a known circuit layout solution grows as the system solves more problems. Floorplanning and interconnection of the subcircuits is example-driven and can be easily biased to favor particular layout styles or technologies.

## II. Learning System

We can define a learning system as a system which can solve the same problem "better" the second time it is presented the problem. "Better" can be taken to mean the solution is better or the time taken to produce the answer is shorter while producing an answer of equal or better quality. We have designed our system to meet both of these conditions; if a new solution is known and judged to be satisfactory, producing it a second time consumes less time. In order to get a better solution a learning system requires feedback on the quality of the solutions it produces. Without some evaluation of its results, it has no guide for improvements. The process of critiquing a layout problem is beyond the scope of this paper, but this task can be performed by a layout expert, aided by circuit simulation and/or device processing considerations. We will assume that a critique system is available and that at the very least, "bad" or problem areas of the layout produced by our system can be identified.

## III. Problem Formulation

One method of looking at the problem of IC layout is to consider it to be a mapping of a set of input parameters into a function to get a solution. This mapping may be described by the following:

### A. Point to Point Mapping

In this scheme, the functional description consists of describing each valid input condition and associating a solution to each of these inputs. As an example, consider the following definition of a function:

Valid inputs: CMOS circuits consisting of circuits for which a "good" layout is known.

Functional mapping: Associate a known layout with the given input circuit.

A disadvantage to this method of describing a functional mapping is that it consumes a large amount of memory. Since knowledge about each particular solution must be retained, considerable redundant information may be included in the knowledge-base; for example, a "good" layout may contain a "good" solution or "good" relative placement for a sub-problem, but no facilities are defined to access this information. If the layout for that sub-problem is known, the associated layout will be maintained elsewhere. An additional disadvantage is the problem of classifying the input. Once the input has been classified, the process of determining a solution is trivial; this process can be implemented with a simple set of pointers from the inputs to the representation of the answers. Accessing an array of these pointers can be performed in a fixed amount of time. However, if the description of a problem circuit has a form which permits multiple equivalent representations of the same circuit, determining the appropriate index for the array of pointers can be difficult. Either time or memory efficiency must be sacrificed to select the correct pointer. Allowing each possible representation to access its own pointer permits the pointers array to become unmanageably large. The alternative is to perform transformations on an input to convert it to a single, recognizable form. Selecting a transformation is related in time complexity to the number of equivalent representations. Applying the transformation is related to the size of the representation. Either time or memory efficiency must be sacrificed to select the correct pointer.

**B. Complete Generalization**

This technique involves defining how the solution can be determined for the entire set of valid inputs. As a numerical example, consider:

Valid inputs: x, a member of the set of all real numbers

Functional Mapping:

$f(x) = \sin(x)$

An advantage to this method of description is that the memory space required is related to the programming necessary to compute the solutions and not the size of the solution space.

A disadvantage is that the time needed to solve a problem is usually related to the size of the input. An additional disadvantage is that a complete generalization may not produce the optimal solution to a problem and determining when this non-optimal performance will occur may be as complex as solving the problem itself. In the worst case, the solution must be obtained by computing and then evaluating the appropriateness of the answer. This results in a time complexity to determine appropriateness which is of the same order as the problem itself!

From the learning system standpoint, this approach is difficult to modify because: 1) the learning portion of the system must contain an understanding of why the algorithm or computation is performed in the manner that it is used, and 2) the learning system must know how the algorithm can be changed and what changes to make.

### C. Partial Generalization

This technique is a middle-ground between the two description techniques examined above. The set of valid inputs is divided into subsets and a complete generalization is associated with each of these subsets. For example:

Valid inputs: x, a member of the set of all real numbers

Functional mapping:

$F(x) = 0$ for all $x < 0$

$F(x) = 1$ for all $x >= 0$

This partial generalization partitions the set of all real numbers into two disjoint subsets and a mapping for each subset has been defined. A second example is:

Valid inputs: x, a member of the set {1, 2, 3, 4, 5, 6}

Functional mapping:

$F(x) = x/2$ for x a member of {2, 4, 6}

$F(x) = (x+1)/2$ for x a member of {1, 3, 5}

Matching a candidate problem to a valid input is generally simple, since it involves a subset operation rather than an exact match. In the first example above, the matching problem is a simple comparison. In the second example, the matching problem consists of determining membership in one of the two sets. If the sets can be ordered with a heap, this membership determination can be performed with a worst case time complexity proportional to the log of the size of the set. Using a hashing function technique [1], the expected time complexity is a constant independent of the set size.

## IV. Past Approaches using Partial Generalization

Several algorithmic and rule-based approaches to automated layout generation which have been proposed to date use a "top-down" approach to partial generalization, as exemplified in references [3,14]. These approaches are very efficient at solving a particular subset of problems, such as CMOS layout and logic which can be described exclusively by boolean equations. In some cases, these algorithms will guarantee an optimal solution. They are limited, however, in that revisions or even minor alterations in layout priorities may involve drastic changes to the basic algorithms. Furthermore, due to the complexity of the algorithms, a designer may not have sufficient algorithm generation expertise to alter the algorithm. As technology changes, the development of new algorithms becomes a necessity; however, developing these new algorithms is difficult and time consuming.

The expert system approach, as presented in references [5,8,13,15] uses rules defined by an "expert" to guide the solution search process. It offers solutions to a much wider range of problems than the algorithmic approach. The trade-off for this broader scope is that an optimal solution is not always guaranteed. Another difficulty is that to revise the priorities of the search process, the set of rules must be revised. As in the algorithmic approach, a designer may not be particularly knowledgeable in the techniques for rule revision. The difficulty of interfacing the

designer's knowledge with the rule-revisor's needs is demonstrated by the efforts that have been expended just defining how an expert (in this case a designer) should be questioned.

## V. Our Methodology

We have chosen a "bottom-up" approach to developing generalizations. "Learning" in this methodology involves recognizing similarities. The level of machine learning used by this methodology is more sophisticated than rote memorization. A system which just memorizes the examples and reproduces them when a problem matches an example's circuit is little more than a solution recording and retrieval system. Our approach allows a large class of problems to be solved by matching similar parts of the problem to good solutions contained within the examples. The result of this process is that problems containing the same circuit elements as the examples can be solved and the quality of the resulting solutions is related to the similarity of the examples (as summarized in the generalized solution graph) and the problem circuit. A basis for our generalization process is contained in references [4,6,11]. The use of searching for a solution by starting from the largest common subset is an extension of the concepts contained in references [5,8,13,15].

Determining if a problem is a subset of the known layout solutions can be a computationally expensive task. If a technique for defining a circuit can be defined which permits only a single, unique way to represent a circuit (such as a canonical form), then a symbol can be associated with the circuit and the membership determination can be performed by hashing, as described in [1], with an expected case time complexity directly proportional to the number of elements in the problem circuit. If a unique representation is not defined, then the problem becomes a sub-graph isomorphism determination. Solving isomorphism with association graphs and clique-finding is described in [2]. These algorithms have a time complexity of at least the square of the number of problem circuit elements.

Before we can use a knowledge-based system, we must define the information we wish to operate upon and define how that information will be represented. In a layout, the positional information is of primary importance. This includes the positioning of the devices relative to each other, significant features of the connecting paths, and the size of the devices in relation to each other. The actual size of the various components may be dependent on the process used to fabricate the circuit and may not directly affect the choice of layouts. In the ideal case, the layout should be scalable to be of use for a variety of fabrication processes. Based on these considerations, we have chosen to represent the knowledge in the system with the following elements:

**Devices** - These elements represent the primitive units used to define the circuit. The primitive units can be transistors, resistors, capacitors, diodes, etc. or they can be combinations of transistors, capacitors, etc. which will be placed as a unit. As such, the information needed are the dimensions of the unit and the location of its terminals. Since we wish to be able to scale the layout, its dimensions need to be specified relative to some standard. Such a standard can be set equal to the minimum-sized feature of a process, and all dimensions can be represented as some multiple of that minimum size.

**Junctions** - The most significant positional information about an interconnection is the number and location of intersections it contains. Since corners and level changes can be considered the intersection of two paths, we treat them as junctions also. All junctions have a minimum area and are defined by the interconnect they join.

**Expandable Connectors** - These represent the straight line connections between devices and junctions. Their significant feature is length. Other parameters, such as width, thickness, etc. are functions of the fabrication process, location on the chip, or the circuit in which they are used. This means that resistors in an analog circuit layout must be treated as devices rather than interconnect. It also implies that length to width ratios for interconnect lines is not considered.

## VI. An Example of Our Methodology

The first step in using a knowledge-base is to provide it an example of a layout. We will use the circuit in Figure 1 to demonstrate the process. Figure 2a shows a corresponding layout. The layout graph, in Figure 2b represents a minimal generalization of this layout, and is developed by identifying the devices, junctions, and straight interconnect runs in the layout and replacing them with the corresponding symbol. If we consider half the width of the minimum-width active area runs as a smallest feature size, then all transistor elements except one have a width of 6 units and a length of 6 units. Minimum-sized metal to metal junctions measure 4 units wide by 4 units long. The other devices and junctions can be defined in a similar fashion. A straight interconnect is abstracted into a variable length expandable connection.

The sample layout can actually be connected to other circuits in more ways than those supplied by the sample. Each active area, polysilicon, or metal junction can have connections to each of their four sides and contact windows can have as many as eight connections. Figure 3 , which shows the additional connections, is called a generalized solution graph and can be used to derive circuits with differing external connections.

Now that we have a knowledge-base, we can begin to solve problems. Consider the circuit in Figure 4. We will use the established knowledge-base to attempt a layout of this problem. To do this, we compare the problem circuit to the generalized solution graph. If we find a match, then no layout needs to be determined - we just need to apply the parameters of the fabrication process to determine the specific locations and sizing of the various components. If as in this case, we do not find a complete match, we must derive a layout. To derive a layout, we consider the portions of the problem circuit for which we already know a layout. Figure 5a shows the largest sub-circuit for which we already know a layout. The unmatched portion of the problem circuit, shown in Figure 5b, is again compared to the knowledge-base to find layouts for the remainder of the circuit. In this case, the database contains a layout for the remainder of the circuit; thus our problem becomes merely one of positioning and interconnecting these two sub-layouts. The placement and interconnection of the two sub-layouts we have chosen is simple, and can be performed by conventional placement and routing algorithms or by a knowledge-based scheme. The two sub-graphs to be connected are shown in Figure 6. The resulting problem solution graph is shown in Figure 7.

The next step of the methodology involves the elimination of unnecessary external connections and the application of process specific parameters to obtain a specific solution. The result of this process for a particular set of external connections is shown in Figure 8.

The final step is to include the information gained by solving this problem into the database. If the derived solution is judged to be satisfactory, then it becomes an example to be incorporated. If deficiencies in the solution are identified, then the improved solution is incorporated into the database. A layout compaction tool could be applied to the derived layout, and would supplement the critique process. When compaction is applied to the example layout, a smaller-area layout results, as shown in Figure 9a. It is very similar to the derived layout and the differences, which correspond to improvements, can be provided as a critique to the system. This critique is generalized into a problem solution graph, shown in Figure 9b, for incorporation into the database.

When incorporating the new information into the database, we search for the largest set of elements which have the same positional relationships in both the generalized solution and problem layout graphs. The parts of the problem solution graph which are not in this common sub-graph are added to the database graph. If adding these parts uses the same connection points on a junction, device, or expandable connector, then the parts are added as an "or" option. This signifies that only one of the parts could appear in a layout. The largest sub-graph common to the example problem and critique is shown in Figure 10. The results of including the critique's features are shown in the expanded generalized solution graph in Figure 11. Note that the "or" branches reside in overlapping positions. Some of the expandable connectors in the figure have been "bent" to add readability to the picture and these bent connectors are the parts of the critique which were not contained in the largest common sub-graph.

**VII. Evaluation of the Solution Layout**

The layout produced by our system is near-optimal. The transistor controlled by signal L could be moved in and the active area to metal connection at the top could be rotated to produce a slightly smaller layout, as shown in Figure 9. These changes could be performed by existing compaction techniques. The positioning of the transistor is a function of the partitioning process. If we had chosen to partition by another criteria, such as fewest number of partitions rather than largest partitions, the positioning of this transistor could have been different. connection, allowing this to be used in future layouts.

Our system incorporates learning as part of its make-up and performs it automatically. A reasonable question to ask is why we would want to have a system solve the same problem again? The answer is two-fold: 1) multiple users could use the same system, and without communications between them, effort could be spent by both of them solving the same problem; 2) the developed layout could comprise a large portion of a larger problem to be solved at another time. Re-deriving this layout is unnecessary if the knowledge can be retained.

## VIII. Quality of Results

This knowledge-based system derives its answers from the samples it is provided. Therefore, the quality of its solutions is dependent on the information it is provided. If the samples provided are "poor" then the solutions it produces will generally be poor, since the solutions will be either generalizations of the samples or generalizations of parts of the samples. On the other hand, if the samples are of "excellent" layouts, then parts of these excellent layouts will be used to derive layouts which are not already known.

The use of parts of the layout examples can be compared to a standard cell system. The "sub-layouts" which this methodology employs are defined by the generalized solution graph. This "library" of sub-layouts can be expanded whenever our system is supplied a sample or a problem (for which it does not already know a layout). Since the interconnection of the sub-layouts is not limited to routing areas and sub-layout placement is not limited to specific locations, the knowledge-based approach will produce layouts which more closely approach custom "hand-crafted" designs. Once a successful layout for a problem has been derived, the time necessary to reproduce that circuit layout is unnecessary - knowledge of that layout is retained. If a problem circuit contains a large subcircuit for which a layout is known, that layout may be selected as one of the sub-layouts and the computational complexity is related to connecting sub-layouts, rather than placing and interconnecting all of the individual devices. By comparison, a standard cell system re-derives the same solution if it is used to solve the same problem a second time.

## IX. Limitations of Methodology

Computationally, the hardest task in this methodology is sub-graph isomorphism. A time complexity of at least the square of the size of the graphs to be compared may seriously limit the maximum size of the database. By incorporating all of the knowledge into a single generalized solution graph, we have reduced the number of isomorphism determinations that must be made. The trade-off is that the generalized solution graph could become very large. We argue that, although the sub-graph isomorphism algorithms do not perform well on large problems, eliminating the examination of redundant information more than makes up for their poor performance in a large graph.

Partitioning of a previously unsolved layout is heavily dependent on the process which selects the sub-layouts. There is no guarantee that the sub-layouts selected from outstanding samples can be combined to produce an optimum layout. Heuristically, good layouts should be composed of good sub-layouts and choosing good sub-layouts as a starting point should lead to an acceptable solution if the heuristic is valid.

Similarity between the generalized solution graph and problem circuits directly affects the size and number of sub-layouts which are considered for connection to produce a solution. If the generalized solution graph contains only small subcircuits common to the problem circuit, then the size of the sub-layouts which can be chosen is small and the number of these sub-layouts which must be connected is large. If the generalized solution graph contains a large subcircuit which is common to the problem circuit, then the layout of the subcircuit can be

considered for inclusion in the sub-layouts used to derive a solution.

The derivation of a solution is admissible (a solution can be eventually found) if the samples contain the same types of devices and a complete set of the connection primitives. However, in the worst case, derivation will consist of placing and routing individual devices and the knowledge-based approach guarantees neither the most efficient layout nor reasonable time complexity. This case implies the samples provide only the most minimal information on layout, which is (hopefully) not a common event. This lack of information also handicaps the other approaches also; if little is known, applicability of a particular algorithm is unknown and few rules can be defined for an expert system based approach.

## X. Conclusions

Use of a learning system implies the system's performance improves with experience. With few good examples to draw upon, the scope and quality of the solutions produced will be limited. If the examples do not contain the minimum information necessary to solve the problem, then a solution derivation may not be possible. However, as experience is gained, the quality and time needed to solve a problem will decrease. With enough information, the performance of a learning system will exceed that of either the algorithm based or expert system approaches.

The rate of improvement of our learning system is directly tied to the quality of the critique of solutions. Minimal feedback on solution quality will limit changes in how problems are solved. Suggestions for a "better" layout cause the changes in how a problem is solved, so improvement is directly related to the quality an applicability of the critique.

Our learning system incorporates changes much more easily than the algorithmic or expert system approaches. The user communicates with the knowledge-base by means of examples which should be familiar to the user and within his realm of expertise. Distilling the important information out of these examples, such as relative placement and significant features of the interconnections, is the task of the system, not the user.

Re-derivation of a solution is not necessary with this methodology. The ability to retain the results of a solution search is included to ensure solving redundant problems consumes little more that the resources necessary to recognize that redundancy. Experience gained individually by multiple users of the same system is available to all.

## References

[1] Aho, A. V., Hopcraft, J. E., and Ullman, J. D., "The Design and Analysis of Computer Algorithms," Addison-Wesley Publishing Company, Reading, Massachusetts, pp. 111-113, 1974.

[2] Ballard, D. H., and Brown, C. M., "Computer Vision," Prentice-Hall, Englewood Cliffs, New Jersey, pp. 365-376, 1982.

[3] Baltus D.G. and Allen J., *"SOLO: A Generator of Efficient Layouts From Optimized MOS Circuit Schematics,"* Proceedings of the 25th ACM/IEEE Design Automation Conference,

Vol. 25, pp. 445-452, Baltimore, M.D., 1988.

[4] Berwick, R.C., *"Learning from Positive-only Examples: The Subset Principle and Three Case Studies,"* in Machine Learning: An Artificial Intelligence Approach, Vol. II, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), Morgan Kaufmann, Los Altos, California, 1986.

[5] Cesear, T., Iodice, E., and Tsareff, C., *"PAMS: An Expert System for Parameterized Module Synthesis,"* Proceedings of the 24th ACM/IEEE Design Automation Conference, Vol. 24, pp. 666-671, Baltimore, M.D., 1987.

[6] Dietterich T.G. and Michalski R.S., *"A Comparative Review of Selected Methods for Learning from Examples,"* in Machine Learning: An Artificial Intelligence Approach, Vol. I, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Tioga, Palo Alto, California, 1983.

[7] Keutzer K., Kolwicz K., and Lega M., *"Impact of Library Size on the Quality of Automated Synthesis,"* International Conference on Computer-Aided Design, pp. 120-123. Washington, D.C., 1987.

[8] Lin Y. S. and Gajski D.D., *"LES: A Layout Expert System,"* Proceedings of the 24th ACM/IEEE Design Automation Conference, Vol. 24, pp. 672-678, Baltimore, M.D., 1987.

[9] Minai A.A., Williams R.D., and Blake, F.W., *"A Discrete Heuristics Approach to Predictive Evaluation of Semi-Custom IC Layouts,"* Proceedings of the 24th ACM/IEEE Design Automation Conference, Vol. 24, pp. 770-776, Baltimore, M.D., 1987.

[10] Mitchell, T. M., *"Generalization as Search,"* Artificial Intelligence, Vol 18 pp. 203-226, 1982.

[11] Mitchell, T. M., *"Learning from Solution Paths: An Approach to the Credit Assignment Problem,"* Artificial Intelligence Magazine, Vol. 3 No. 2, pp. 48-52, Spring 1982.

[12] Mostow J., *"Toward Better Models of the Design Process,"* Artificial Intelligence Magazine, Vol. 6 No. 2, pp. 44-56, Spring 1985.

[13] Odawara, G., Hamuro, T., Iijima, K., Yoshino, T., and Dai, Y., *"A Rule-Based Placement System for Printed Wiring Boards,"* Proceedings of the 24th ACM/IEEE Design Automation Conference, Vol. 24, pp. 777-785.

[14] Shiraishi, Y., Sakemi, J., Kutsuwada, M., Tsukizoe, A., and Satoh, T., *"A High Packing Density Module Generator for CMOS Logic Cells,"* Proceedings of the 25th ACM/IEEE Design Automation Conference, Vol. 25, pp. 439-444, Baltimore, M.D., 1988.

[15] Steele R.L., *"An Expert System Application in Semicustom VLSI Design,"* Proceedings of the 24th ACM/IEEE Design Automation Conference, Vol. 24, pp. 679-686, Baltimore, M.D., 1987.

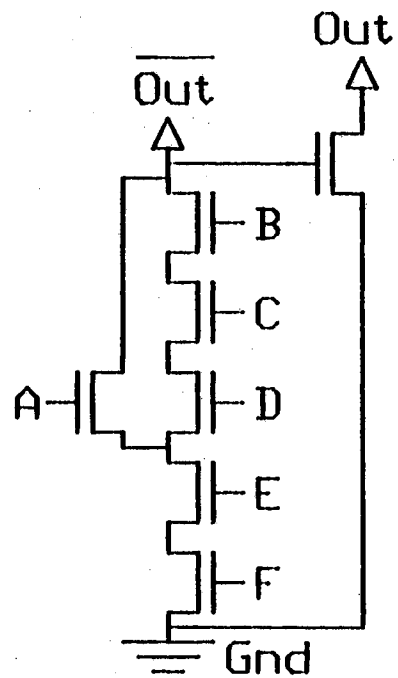
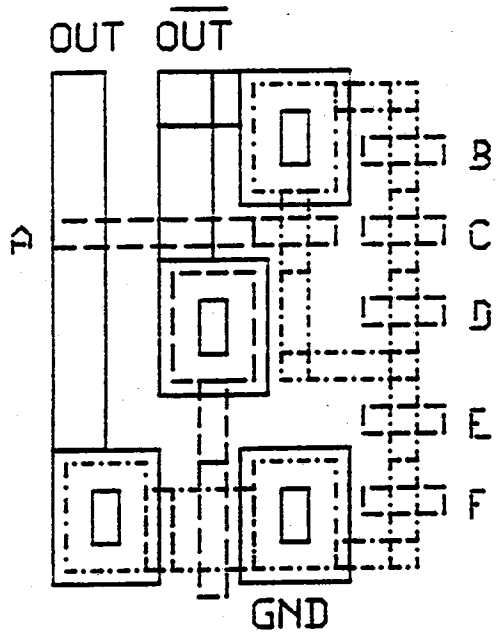
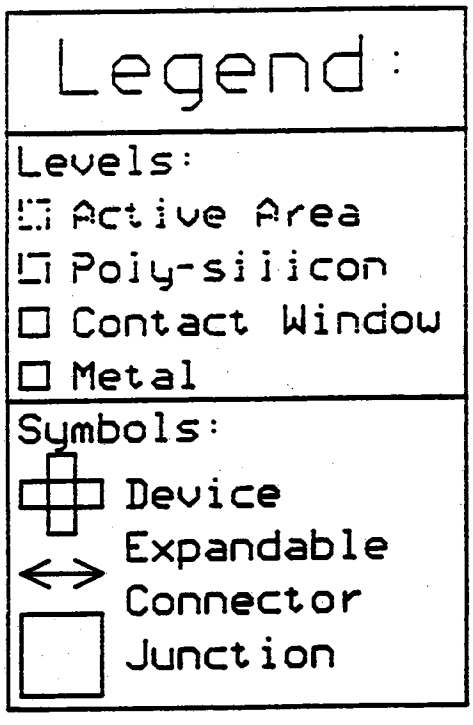
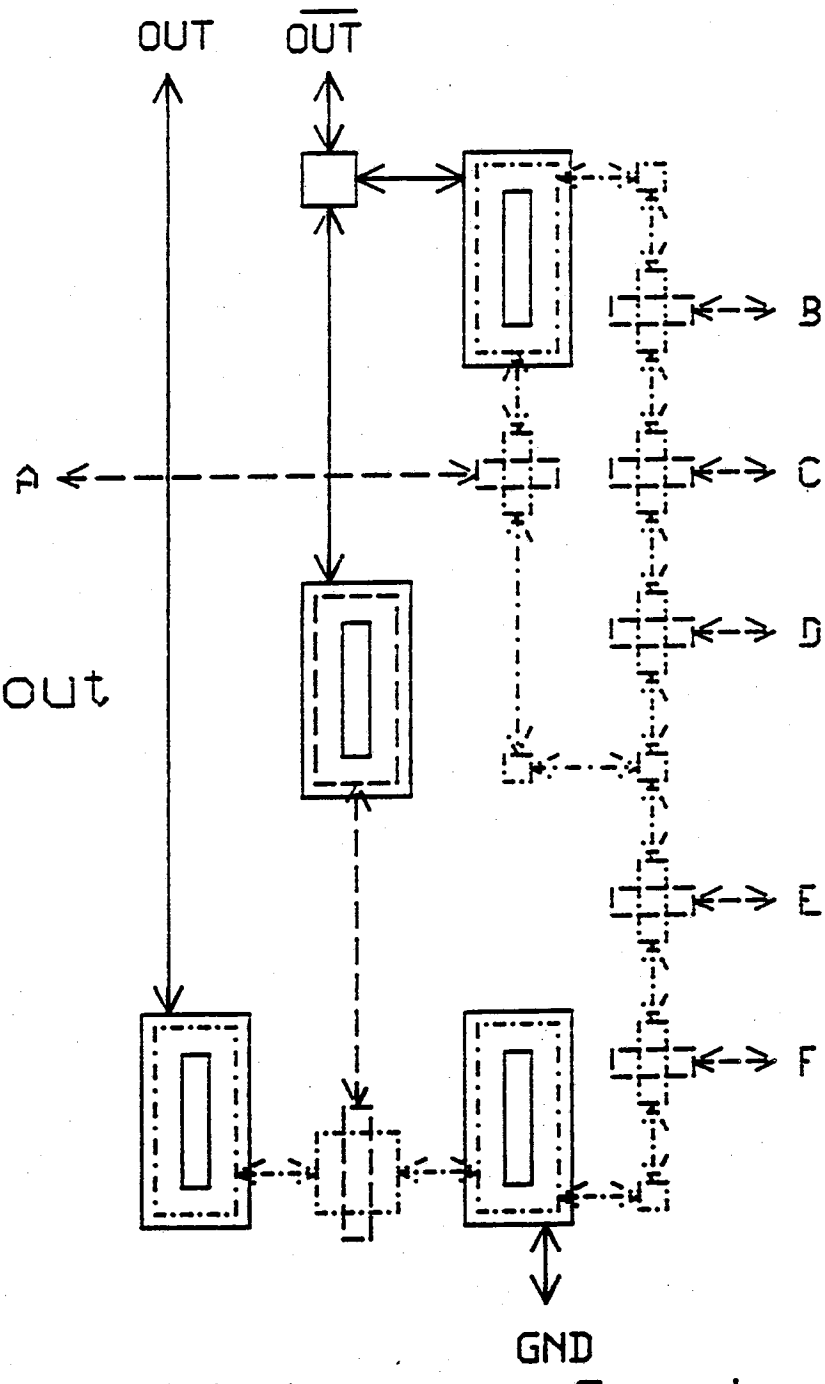



Figure 1. n-channel portion of a complex CMOS gate




(a) Sample Layout




(b) Layout Graph

Figure 2.

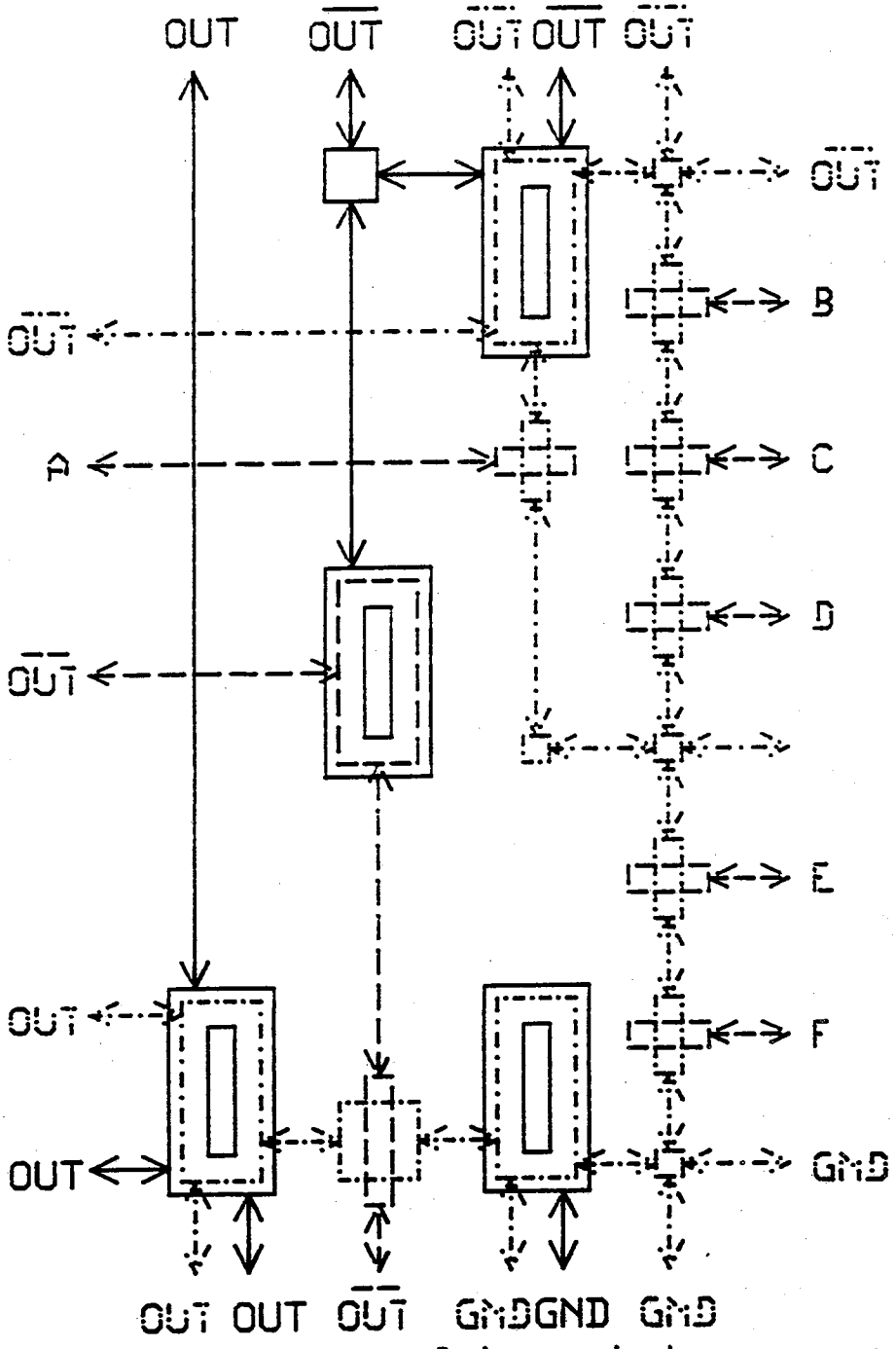



Figure 3. Generalized Layout Graph

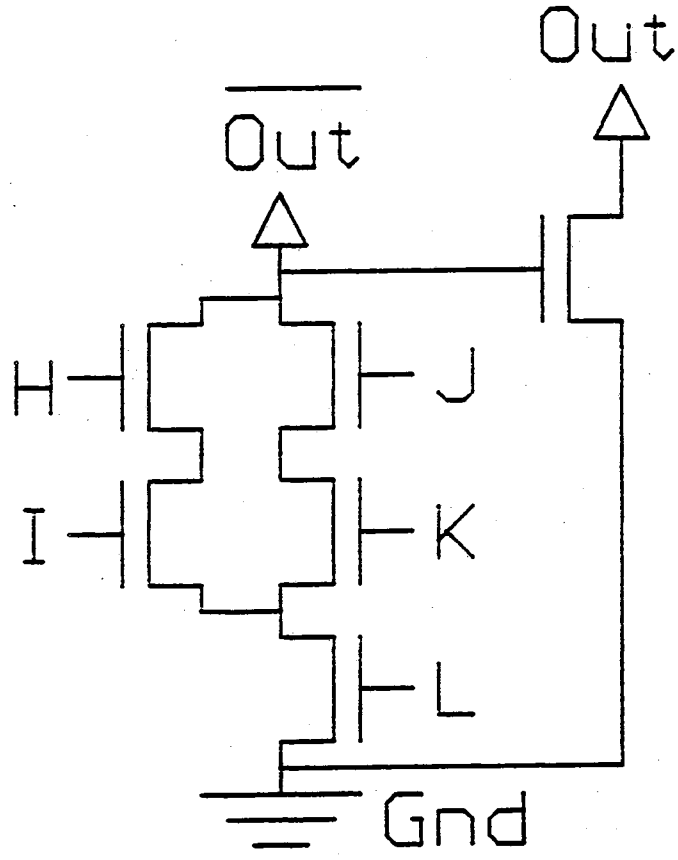
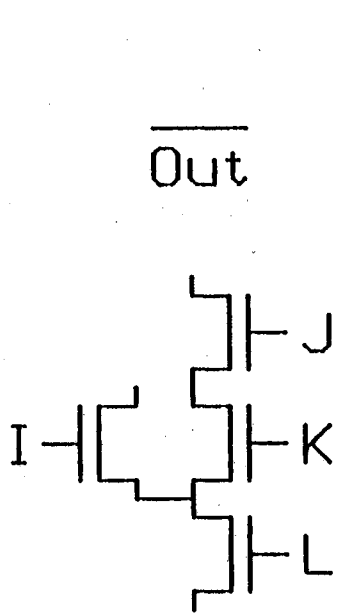
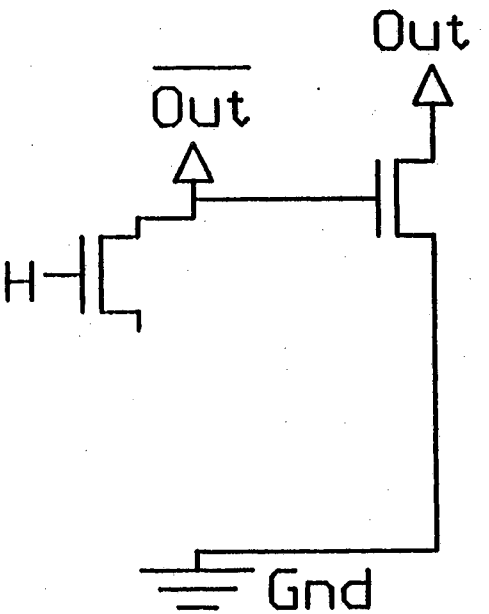



Figure 4. Sample Problem

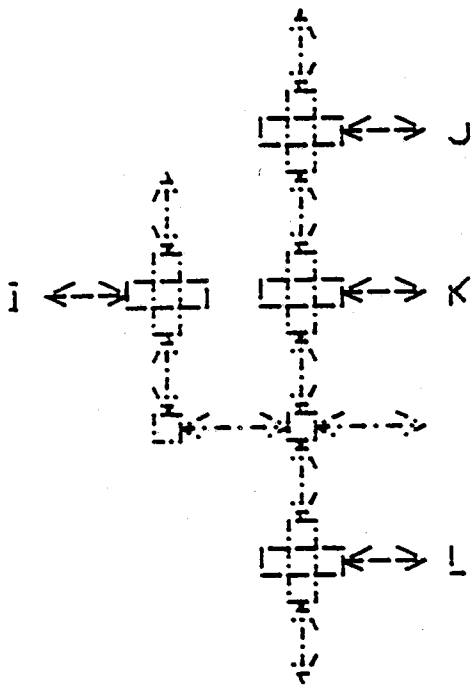



(a) Largest Subset

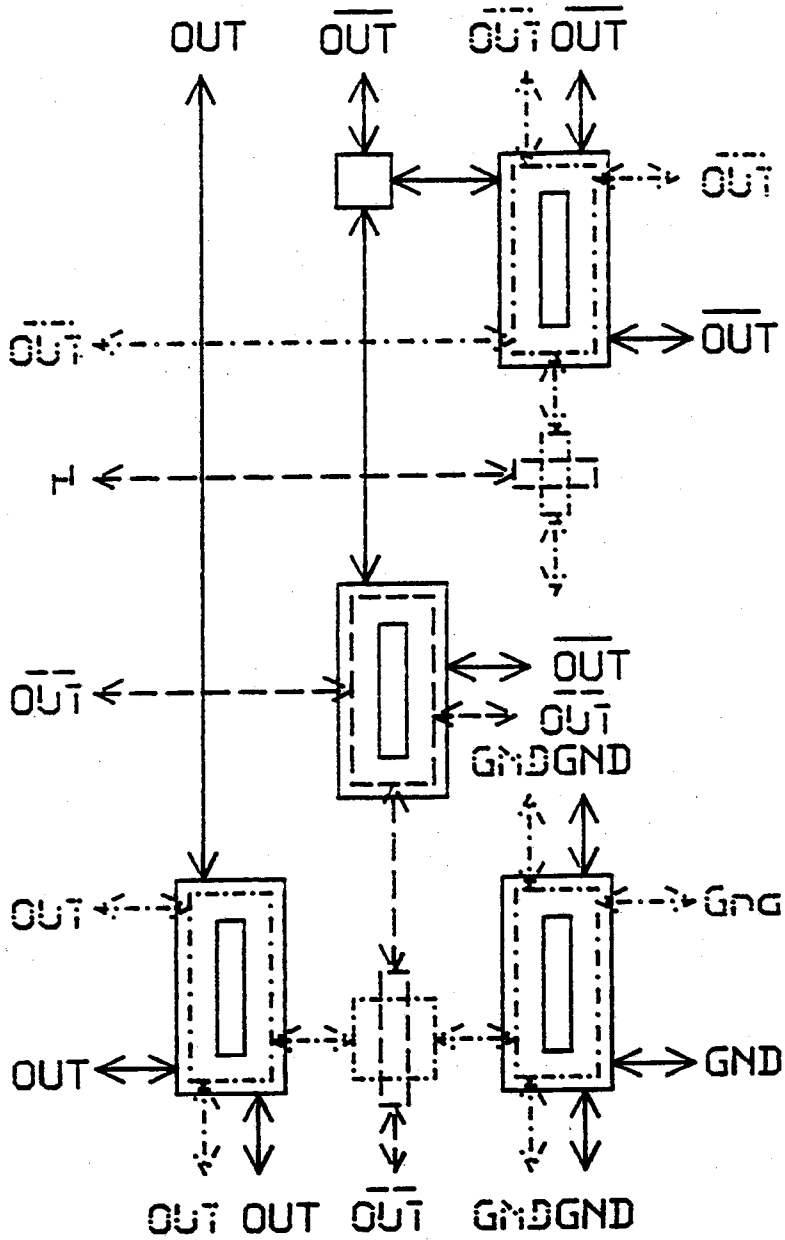



(b) Remaining Subcircuit
(Also a Subset)

Figure 5.




(a) Largest Sub-layout Graph




(b) Other Sub-layout Graph

Figure 6.

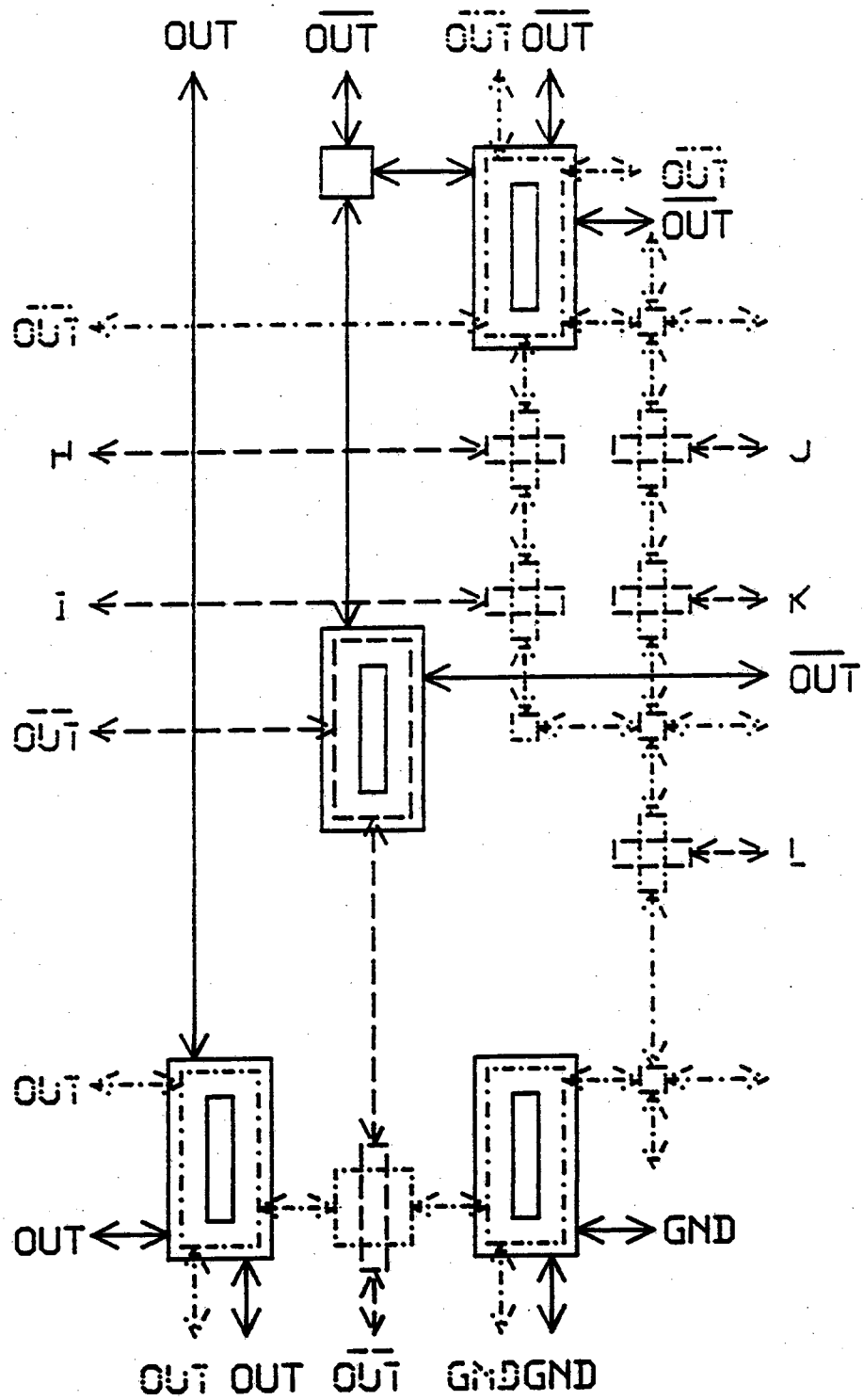



Figure 7. Problem Solution Graph

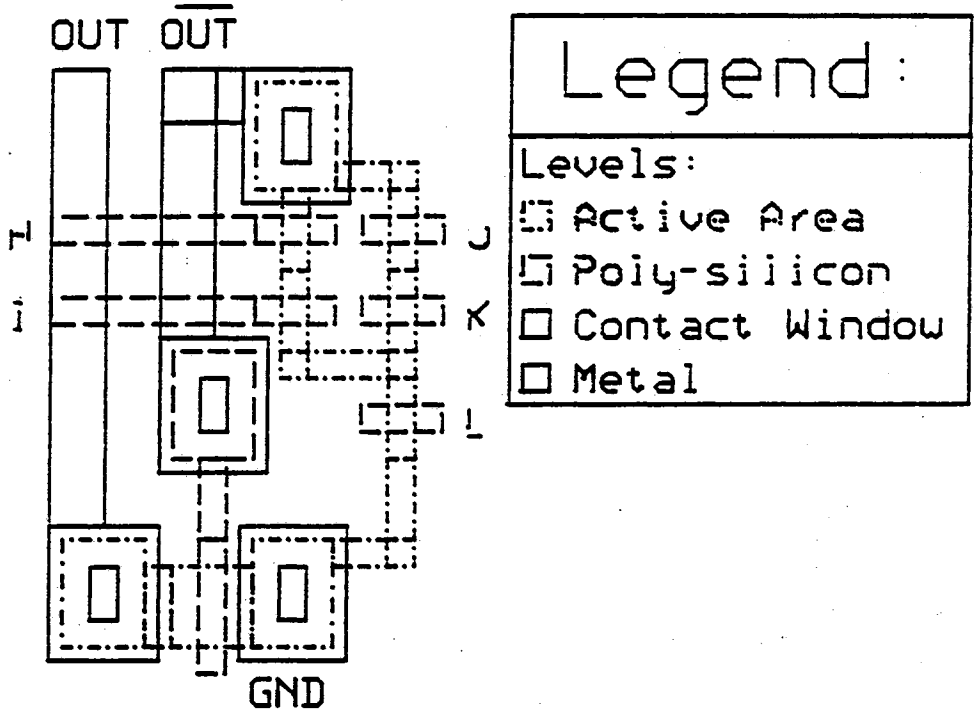



Figure 8. Solution Layout

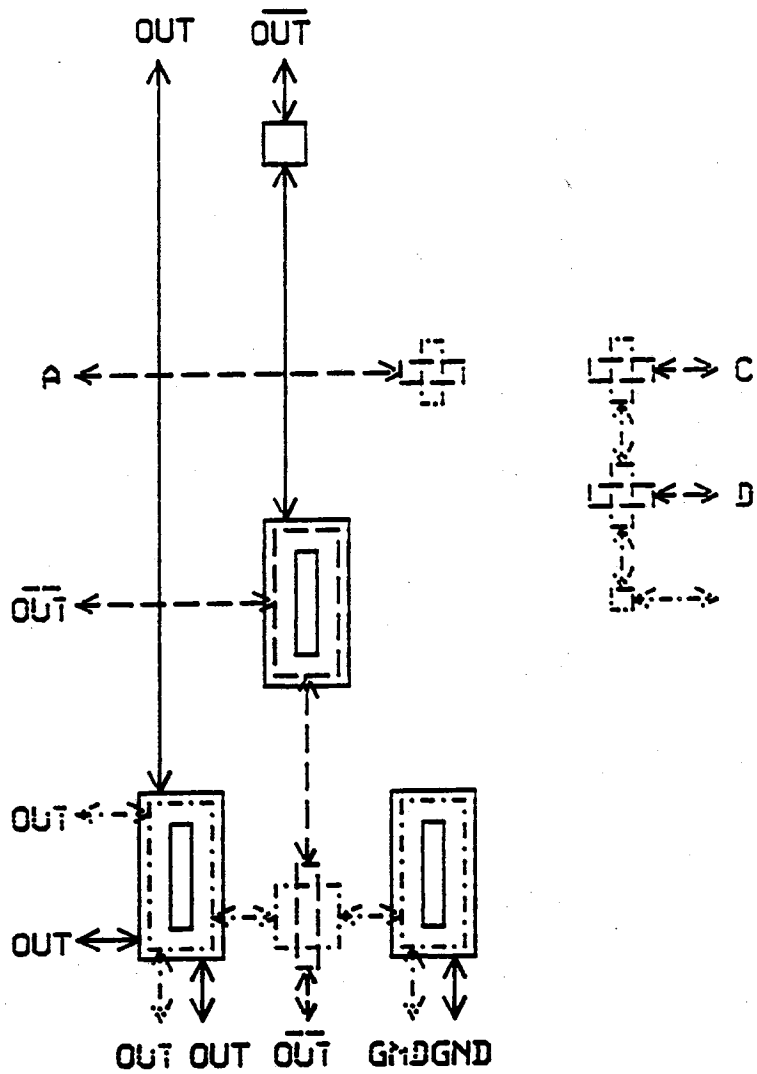



Figure 10. Elements in Common Between the Generalized Solution Graph and the Problem Solution Graph

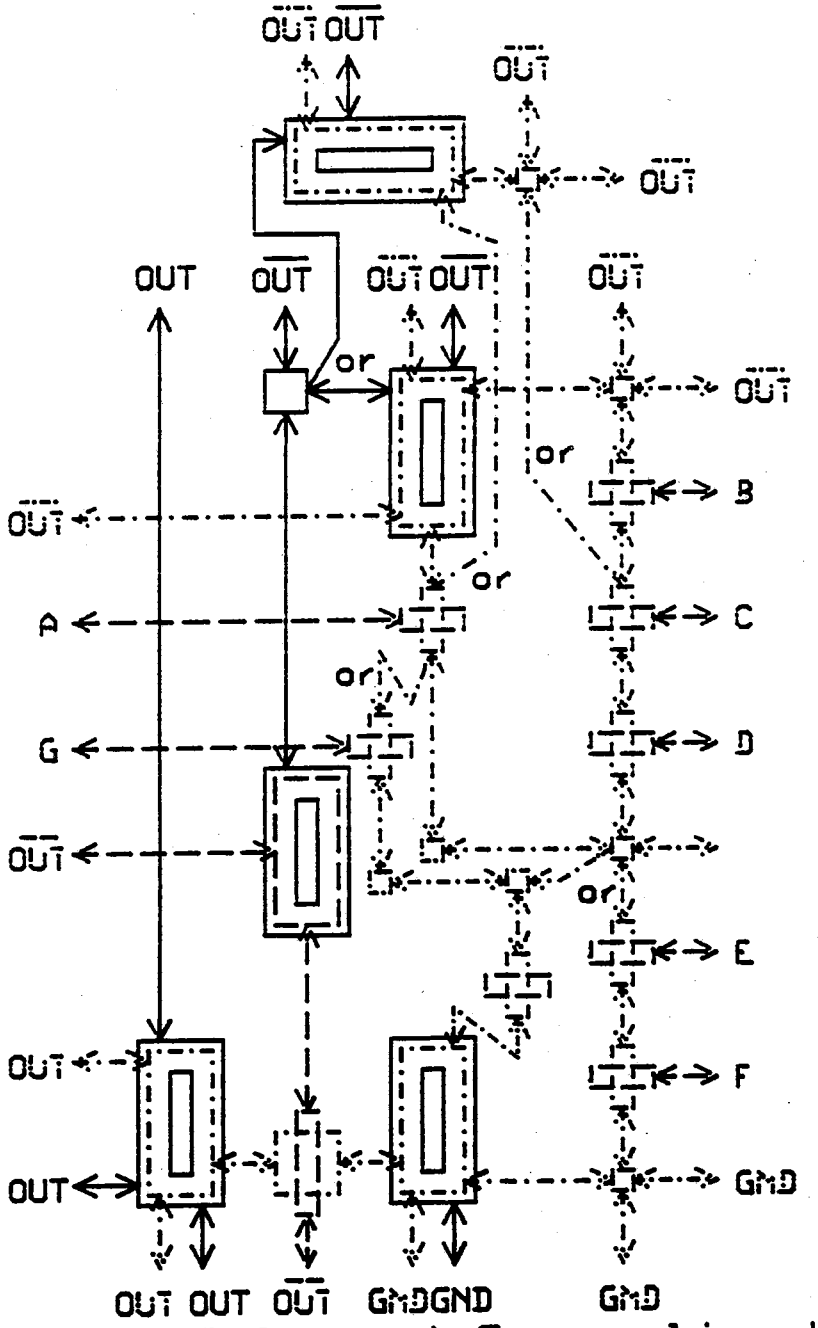



Figure 11. Enlarged Generalized Layout Graph