From: AAAI-83 Proceedings. Copyright ©1983, AAAI (www.aaai.org). All rights reserved.

TALIB: An IC Layout Design Assistant

Jin Kim and John McDermott

Departments of Electrical Engineering and Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

Abstract

This paper describes a knowledge-based system for automatically synthesizing integrated circuit layouts for NMOS cells. The desired cell layouts are specified in terms of their general structural and functional characteristics. From these initial specifications, the system produces correct and compact cell layouts. The system performs this task by generating plan steps at different levels of abstraction and opportunistically refining each plan step at one level to more specific steps at a lower level. Although the implementation of this system has focused on NMOS technology, the techniques used are not restricted to that technology.[1]

1. Introduction

In recent years there has been a growing interest in developing systems that automatically synthesize integrated circuit layouts from high-level functional descriptions. The motivation for developing automatic design synthesis systems is three-fold: reduction of design costs through elimination of design errors, efficient exploration of the design space, and reduction in design time by allowing designers to work at a higher level of abstraction. This paper describes a system, called TALIB, that assists with the part of the IC synthesis task known as cell layout. The program is implemented in OPS5, a general-purpose production system language [Forgy 77] [Forgy 81]. TALIB accepts a description, in netlist form, of the schematic of the NMOS digital circuit to be laid out as input and produces the description of the mask geometry as output.

A typical integrated circuit is implemented on a silicon wafer by creating, through a variety of fabrication techniques, layers of different substances in geometric patterns on the wafer surface. The superimposed layers of conducting, semiconducting, and insulating materials define the electronic components by interacting through different layers. The task of laying out an integrated circuit involves determining what patterns on each layer will create the desired components and interconnections. For example, in the NMOS technology that TALIB uses, a transistor is created whenever a region on the polysilicon layer overlaps a region on the diffusion layer. The goal of the layout task is to put as much circuitry as possible in as small an area as possible. Of course the resulting circuitry must work and satisfy the boundary constraints on the layout.

[1]This research was funded through a fellowship from the Xerox Corporation and through National Science Foundation grant ECS-8207709. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Xerox Corporation or the National Science Foundation.

In order to manage the complexity of large chip design problems, it has proven useful to take advantage of the hierarchical structure inherent in most digital circuit designs [Mead 79]. By utilizing the hierarchical structure, the complexity of the chip design problem can be managed by partitioning the original problem into less complex subproblems at lower levels in the design hierarchy. For example, a chip floor plan that partitions the layout into functional areas and constituent cells can be prepared at the top level. With the chip plan as a guide, each functional area can then be constructed from cells and devices in a hierarchical manner. The functional areas are subsequently merged and fine adjustments made to both the cells and the chip floor plan until the cells are bound together tightly.

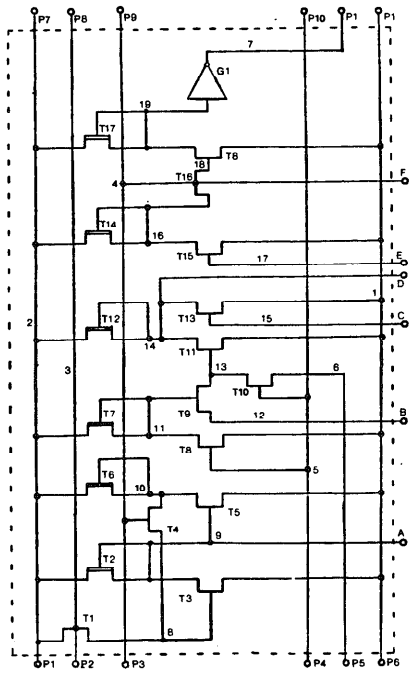

Figure 1-1: NMOS Digital Circuit Schematic

There are numerous problems associated with automating the hierarchical layout design methodology. The focus of this paper is on the problem of laying out the cells. The cell layout problem is formulated in terms of a schematic of the circuit to be laid out and the boundary constraints on the layout of the cell. For NMOS chip design, the circuit schematic typically

contains a mix of logic gates and transistors as shown in Figure 1-1; the boundary constraints describe the size of the cell, its shape, the aspect ratio, and the ordering of i/o signals along the cell sides. The boundary constraints reflect the characteristics which the cell's layout must have in order to fit into the overall chip layout. Solving the cell layout problem involves generating geometric layout patterns for the circuit schematic which satisfy the boundary constraints as shown in Figure 1-2.

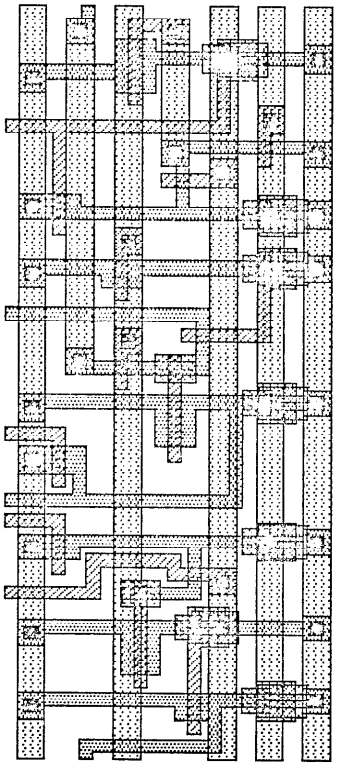

Figure 1-2: NMOS Layout

2. TALIB

The task of layout generation is that of placing box-like modules on a plane bounded on the X and Y axes, and connecting them by a set of wires according to a given set of rules. In the process of planning a layout, the designer must generate intermediate steps based on expectations. For example, in placing a subcircuit on a layout surface, the designer must make assumptions about how it will be laid out in order to analyze its impact on such global parameters as usage of metal channels and total area; however, the task of actual layout of the subcircuit cannot be started until its placement relative to other subcircuits is known. This facet of tentative reasoning introduces uncertainty into the design environment. Since such uncertainty is inevitable in bin-packing problems, TALIB is implemented as a planning system.

In addition to handling uncertainties in the design environment, plans are useful in distinguishing between important considerations and details. By utilizing knowledge at higher levels of abstraction, TALIB can eliminate, at an early stage, those search paths that are not very promising [Sacerdoti 77]. Further, by using plans to control focus, TALIB can introduce global information into the layout generation process. Without the plan structure, TALIB would tend to search in a depth first manner by relying only on local information.

Although TALIB develops and refines plans around objects at various levels of abstraction, most of its knowledge is based on empirical rules that human experts utilize. Because TALIB lacks any deep understanding of the task domain, it cannot explain why its rules are valid.

2.1. Input Description

The input to TALIB consists of two parts. The first part describes the circuit components and their interconnections. A circuit is made up of devices and signal nodes. A device is any of the components that are normally present in a digital circuit such as NOT-gates, NOR-gates, and transistors. Each device has two or more terminals by which it is connected to the rest of the circuit. Two device terminals are connected to each other through a common signal node. The transistor sizes in the circuit schematic can be specified explicitly or default values can be generated by TALIB.

The second part of the input describes the topological and geometric requirements around the outside boundary of the circuit layout. The topological information includes the order in which external connections must appear on the cell boundary. This requirement can vary in detail from a specification of just the side of the cell on which a signal is to be made available, to a specification of the exact ordering which the signal node must have relative to other signal nodes on that side. The description of each i/o signal can also include the layer on which the signal must be implemented. The geometric information includes the specification of the physical parameters of the cell layout; this can be stated in terms of the upper bounds on the dimension of the cell layout or in terms of the cell's aspect ratio. The combination of the topological and geometric descriptions of the cell layout make up the boundary constraints of the cell layout problem. The output of TALIB is the description of the layout geometry in CalTech Intermediate Form (CIF) [Mead 79].

2.2. Layout Models

TALIB is capable of limited spatial reasoning. In the degenerate case when very little knowledge can be brought to bear on a particular region of the layout, TALIB expands the relevant transistors and their interconnections in terms of geometric primitives (e.g. rectangles) within the locality. Inside the expanded region, TALIB relies on a modified form of the Lee expansion routing algorithm [Lee 61] to continue the layout process. However, the search space associated with even a small circuit can be very large when it is handled at the geometry level. If all the details of the search space were attended to at this level, the combinatorics would enable TALIB to solve only rather simple problems. Therefore, TALIB represents the state of the layout in terms of constructs that deal with subcircuits rather than rectangles. By working with abstractions of the layout and dealing with geometric primitives only when necessary, TALIB is able to solve non-trivial problems efficiently.

The design constructs used by TALIB are built around subcircuits that are commonly found in digital NMOS circuits. These subcircuits are composed of two to six transistors, have a small number of satisfactory layouts[2], and are usually used many times in a design. For each subcircuit, TALIB creates a collection of objects to represent the characteristics of the subcircuit instances at different levels of abstraction. For example, a subcircuit is represented as a single object with its estimated layout area as one of its attributes at one level of abstraction, as a set of partially ordered objects that each represent a signal terminal of the subcircuit at a lower level of abstraction, and as a collection of geometric primitives at a still lower level. Associated with each of the objects describing a subcircuit is a set of rules for updating the attributes of the object.

In addition to the subcircuit based constructs, TALIB maintains a map of the layout surface in terms of unused areas. Initially, the partitioning of the layout surface corresponds to the number of subcircuit instances in the design. As the design task proceeds, additional unused areas will be generated to reflect opportunities for compacting the layout. The adjacency relationship between unused areas reflects a particular topological style TALIB selects to carry out the layout. The topological style selected depends on the initial boundary constraints and on the complexity of the circuit. The task of placing components corresponds to that of mapping subcircuits or geometric primitives to one of the unused areas.

One example of a subcircuit-based design construct is the *circuit-layout* model. For each *circuit-layout* model, there are a set of rules that fill in and update the attribute fields of the model. The attribute fields include the estimated X and Y dimensions of the layout, the geometric components required to realize the layout, the spatial relations among the geometric components, and the orientation of the components in the model. Some of the attributes of a *circuit-layout* model for an inverter are shown below.

Inverter
{comment: this is a circuit-layout
model for an inverter}
Name
Status
X-dim
Y-dim
:
Input-1-type
Input-1-signal
:
Pullup-type
Pullup-length
Pullup-width
Pullup-orientation
Pointer-to-pullup
:
Pullup-pulldown-y-spacing
:

[2]A suitable layout for a subcircuit will vary with the particular set of boundary constraints

In addition to the above objects for describing the state of the design locally, there are global descriptors that reflect the general state of the layout. These include objects that specify the estimated space for metal channels under the given set of boundary constraints, the number of unused channels, the maximum X and Y bounds, and the amount of space remaining in the X and Y dimension.

2.3. Knowledge Organization

TALIB performs the layout design task by utilizing a large knowledge base of design heuristics encoded as rules. The rules embodying the task specific knowledge can be roughly classified into two categories. The bulk of TALIB's rules are organized around the concept of subtasks; this type of organization has proven to be convenient for capturing knowledge that has a relatively narrow scope of applicability [McDermott 82]. The rest of TALIB's rules are demons that select subtasks, detect their completion, and detect constraint violations. Demons have been useful in other task environments requiring the system to react quickly to changing situations [Haley 83].

The subtasks known to TALIB roughly correspond to the steps in the plan developed during the design process. The subtasks are invoked in a hierarchical fashion with those handling the more abstract goals being expanded in terms of the more specific, lower level, subtasks. At the lowest level, the primitives in the subtask hierarchy deal with generating the layouts for commonly occuring subcircuits and with removing unused space between adjacent subcircuit layouts. At a higher level of abstraction, subtasks deal with placement of subcircuits relative to one another and with exploring promising areas on the layout surface. Most of the subtasks are familiar to human experts, but a few, like those dealing with backtracking, are almost never mentioned. The version of TALIB reported in this paper knows how to perform about 100 subtasks.

TALIB's other rules, the demons, allow global knowledge to be brought to bear whenever it is relevant. These rules have four different functions:

- Classifying design situations. A typical rule with this function generates adjacency relationships among subcircuits based on the functional nature of each subcircuit and on the interconnection characteristics of the subcircuits.

- Creating, intantiating, and updating plan steps. A typical rule with this function sets up precedence requirements among plan steps.

- Propagating constraints from one subproblem to another. A typical rule with this function propagates distance relations between signal terminals of one subcircuit and those of another.

- Detecting the completion of tasks and constraint violation in the design-state. A typical rule with this function detects the violation of a fabrication process spacing rule for geometric primitives.

Most of the knowledge at lower levels of the subtask hierarchy is reliable (i.e. if the current state is on a solution path, the state that results from applying such a rule is almost certain to be on the solution path). The bulk of the knowledge at higher levels, however, can only be applied with limited confidence that the result will lead to a solution (e.g. knowledge of how to cluster and place subcircuits). As a consequence, TALIB has to be able to backtrack.

2.4. Control Cycle

The reasoning process is quite basic, with the selection of subtasks being guided by a set of domain-specific control rules. The system reasons forward from the known facts and its knowledge about different layout models to develop a plan structure; as soon as a plan, reflecting the global interaction among subproblems, is developed at one level, the plan steps are expanded in terms of those at a lower level of abstraction in an opportunistic fashion [Hayes-Roth 79]. Since the subtask associated with a particular plan step is instantiated by TALIB whenever sufficient information is locally available, TALIB occasionally develops isolated planning islands that are not on the solution path. This type of thrashing is caused by premature introduction of default information and is minimized through the use of domain-specific control rules. The planning process continues until a solution or a violation of a boundary constraint is detected. If the later occurs, the system backtracks to undo some design decision and attempts an alternate approach; this involves maintaining and analyzing dependency records to trace inconsistencies back to the appropriate inferential steps [Doyle 79].

TALIB's major design activities are:

- Selecting design constructs on the basis of boundary constraints.
- Partitioning the layout surface into zones and, based on a particular layout style, characterizing the spatial relationships among the zones.
- Partitioning the circuit netlist in terms of known topological groupings.
- Placing topological groupings of subcircuits in different zones of the layout surface.
- Refining design decisions within a particular locality and propagating design decisions to other parts of the partial design.

Although, there is a partial time ordering among certain parts of the above design activities, it is not possible to generate an a priori sequencing plan for all the design activities. This is because most of the activities listed above are dependent on specific design situations.

2.5. Status of TALIB

In large chip designs, the bulk of the area savings are due to the simplified global routing between cells rather than as a result of particularly space efficient cell layouts. Based on this observation, we elected to trade off reduction in design time (less search) against some inefficiency in cell layout space as a goal in developing TALIB. The version of TALIB reported in this paper reflects this goal by basing the bulk of its design activity around concepts, the subcircuit clusters, that are at a higher level of abstraction than the geometric primitives. As a result, though TALIB typically generates topological plans that are as good as those produced by human experts, the final area of the cell layouts is 10 to 35 percent greater than those of human designers. Subsequent versions of TALIB will improve upon this figure,[3] but we do not expect TALIB to ever produce cell layouts that are consistantly superior to those produced by human experts.

The current version of TALIB consists of about 1200 rules with about 940 of these associated with specific subtasks. It has been used to generate a variety of cell layouts for each of a dozen circuits. The most complex of these circuits consists of 86 transistors with an ordering specified for the signals at the cell boundary. The circuit shown in Figure 1-1 is part of a multiplier slice and represents one of the simpler circuits laid out by TALIB. Since the boundary constraints on the circuit in Figure 1-1 did not severely restrict the size of the layout, TALIB was able to produce the layout shown in Figure 1-2 with fewer than 2000 rule firings. Figure 2-1 shows some of the subcircuit clusters generated by TALIB during its planning process. We are currently refining TALIB's rule base and its user interface before releasing it for evaluation to a small community of IC designers at CMU. Eventually, TALIB is intended to be part of CMUDA, a hierarchical design automation system for generating complete IC chips from behavioral level specifications [Parker 79] [Joobbani 82].

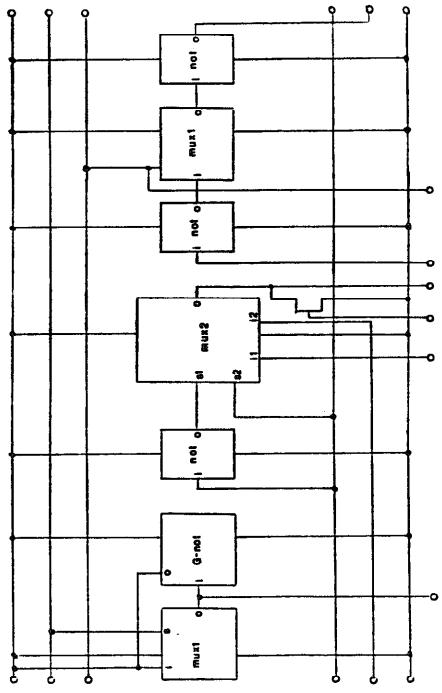

Figure 2-1: Subciruit Groupings

[3] Because a great deal of the domain knowledge involved in chip design is closely coupled to the individual IC fabrication lines, this knowledge tends to be very dynamic. In order to make the knowledge acqusition task for the initial version of TALIB reasonably tractable, the layout domain was constrained to follow the Lambda-based design rules reported in [Mead 79]. Although using these rules allows for effective decoupling from any one particular fabrication process, it does so only through the use of a more conservative design rule set that generally results in larger layout areas.

3. Conclusions

A knowledge-based system for automatically synthesizing cell layouts has been described in the context of a hierarchical chip design environment. The cell layout task contains a number of NP-hard problems. Unless the layout is based on a tightly constrained topological style, such as PLAs or gate-matrices, the number of alternatives that must be explored before a candidate solution can be declared acceptable is ordinarily very large. Based on the observation that efficient chip layouts can be produced without locally optimizing the layouts of the constituent cells, a system for planning design steps around abstract layout concepts has been developed.

TALIB has two basic strategies which help it limit the amount of search required: (1) It generates plans and then refines them at several levels of abstraction; such planning allows it to identify and tackle the least tractable problems first. (2) Its plans are generated on the basis of a variety of features of the circuit to be laid out; TALIB has a significant amount of knowledge which enables it both to quickly generate an abstract plan that is likely to be appropriate and to refine that plan to some extent before exploring alternatives.

Acknowledgements

We wish to acknowledge Dan Siewiorek at Carnegie-Mellon University for the inspiration, ideas, and advice he has provided. Also, we would like to thank Rick Barth and Don Scharfetter of Xerox PARC and Art Krahmer, Bill Mills, Ed Snow, Doran Wilde, and Randy Young of Intel Corporation for their help in developing TALIB's knowledge-base.

References

[Doyle 79] J.Doyle.
A truth maintenance system.
Artificial Intelligence 12:pp231-272, 1979.

[Forgy 77] C.Forgy and J.McDermott.
OPS, A Domain Independent Production System Language.
In *5th Joint Conf. Artificial Intelligence*. ACM, 1977.

[Forgy 81] C.L.Forgy.
OPS5 User's Manual.
Technical Report CMU-CS-81-135, Carnegie-Mellon University, 1981.

[Haley 83] P.Haley, J.Kowalski, J.McDermott, R.McWhorter.
PTRANS: A rule-based management assistant.
Technical Report in preparation, Carnegie-Mellon University, 1983.

[Hayes-Roth 79] B. Hayes-Roth and F. Hayes-Roth.
A Cognitive Model of Planning.
Cognitive Science 3:pp275-310, 1979.

[Joobbani 82] R.Joobbani.
Knowledge-Based Chip Planning System.
PhD Thesis Proposal, Carnegie-Mellon University, 1982.

[Lee 61] C.Y.Lee.
An algorithm for path connections and its application.
IRE Trans. Electron. Comput. :pp346-365, September, 1961.

[McDermott 82] J.McDermott.
R1: A Rule-Based Configurer of Computer Systems.
Artificial Intelligence v19(9):pp39-88, September, 1982.

[Mead 79] C.Mead and L.Conway.
Introduction to VLSI Systems.
Addison-Wesley, 1979.

[Parker 79] A.Parker, D.Thomas, D.Siewiorek, M.Barbacci, L.Hafer, G.Leive, and J.Kim.
CMU Design Automation System: An Example of Automated Data Path Design.
In *16th Design Automation Conf.*. IEEE, June, 1979.

[Sacerdoti 77] E.D.Sacerdoti.
A Structure for Plans and Behavior.
Elsevier, New York, 1977.