IC design—Misery or magic

by K. J. LOOSEMORE
Pompador Ltd.
Hertfordshire, England

Ever since integrated circuit designers began to put thousands of transistors onto a single chip the cost, in terms of human labor, required to lay out the circuit has been extremely high. If we analyze the layout task we find, in general, that two almost distinct phases are involved. First the designer will design the basic building blocks for the circuit and connect them together. The former of these two phases of design requires considerable expertise and experience, making efficient automation extremely difficult. The second phase, however, only requires lower grade labor, since the efficiency of this part of the layout in terms of size and electrical characteristics is usually less critical. The second phase, none the less, requires a considerable proportion of the design time and it is this fact, together with the need for it to be carried out from scratch for each new design and its lower criticality factor that make it a candidate for automation.

The MAGIC suite of programs has been developed at the UNIVERSITY of EDINBURGH both to address the interconnection problem and to attempt to provide some structuring facilities for the more efficient design of integrated circuits.

The system consists of several programs centered around a main automatic/interactive layout generator. It is this main layout generator which is of most interest but it is worthwhile to give a general description of the rest of the suite in order to see how the system fits together.

The designer begins by taking his circuit diagram and splitting it up into a number of functional modules (e.g., a stage of a shift register or part of a decoder). Specifications for these functional modules are then written in the system's input language. At this stage there are only two requirements, first the rectangular outline of each module must be specified and secondly the positions of the interconnect points (called "pins") to the modules, must be defined. It is not necessary at this point to have the logic internal to the module defined. This is required only at the final output stage of the suite.

Bonding pads around the outside of the chip are designed in a similar manner, however, their positions around the chip are vaguely specified (only side allocations and order are required). These definitions, once produced, would normally be kept in a library file (Refer to Figure 1) and made available for future designs.

After specifying the logic modules the details of their interconnection are defined. For each connection net a list of connections is given and, in order to minimize errors at this point, it is also necessary to include a list of all defined connection points which for this particular design are not connected. This feature allows the program to detect accidental non-connections.

In addition to the definition of the circuit the conductor width must be specified and also an estimate given for the final chip size. When the information is complete the input file and library file are compiled and an internal system file generated from which the layout program will proceed. Because several attempts are normally made to iterate to the best layout, this precompilation phase removes the need for the data to be re-checked for every iteration.

The program builds a chip layout in a series of successive levels working from one edge (bottom) of the chip progressively towards the opposite edge. At each level, an area (slot) at a time is locally optimized; components are chosen for each slot which are most closely connected to the existing portion of the layout and to any other components already selected for the current slot. Placement of components and routing of conductors takes place in parallel allowing a global view of the problem to be used and making it possible to organize the layout very efficiently.

The program begins by taking the estimate of the chip size and assuming a square chip, calculates the side length of the chip. The bottom edge of the chip is then laid down and one of the groups of pads specified in the input file is assigned to this side. The set of pads is allocated by finding the component most closely connected to all the pads then choosing the set of pads most closely connected to this component. This means the first group of pads will, as near as possible, all have a target connection in the first slot thus reducing the amount of routing to successively higher slots until the connection is made.

Once the first group of pads have been allocated, the other groups of pads are assigned to other sides of the chip in the order specified in the input language.

Layout in earnest can now begin. The layout proceeds from this point in a series of rectangular slots, each defined by the lowest level reached in the already completed layout and limited horizontally by obstacles protruding up at either end of the horizontal plateau. (see Figure 2.) Clearly the
first slot is the width of the entire chip and successive slots are generated in much the same way that rectangular stones of varying size are laid when building a stone wall.

An attempt is made to optimize the use of the area in the slot both by choosing the best set of components to fit the area of the slot and by minimizing the amount of interconnection required. In filling the slot with components it is possible to calculate the total width required for a given set of components. This is simply the sum of the widths of the components (with appropriate orientations) and the width required for routing conductors vertically between the components. This latter factor consists of the space required for conductors going straight up through the slot without connecting to anything and the space required for connections to side pins of components. If the order and orientation of the components is well chosen this space required for vertical routing can be minimized. For instance if a net connects to two or more pins in the slot there are two possibilities; (1) the pins are in different intercomponent gaps, in which case two or more vertical conductors are required and (2) the pins are in the same gap, in which case only one vertical conductor is required to service both of them (see Figure 3).

There are several other features which influence the orientation and ordering of components in a slot. Firstly, it is more efficient, in general, to have components oriented with their largest sides horizontal. This tends to produce a few large plateaus on which to build further slots rather than several small ones thus increasing effective use of area due to reduced fragmentation. Secondly the orientation of components is a critical factor in determining the number of conductor cross-overs required. For electrical reasons it is important to have as few cross-overs as possible so, within the constraints set by the previous requirements the components is a critical factor in determining the number of cross-overs in their connections with the already existing part of the layout. The third factor on which placement depends is the ordering of the components in the slot.

An attempt is made to minimize the horizontal routing required in the slot by taking this into consideration when ordering and orientating the components. There is obviously a trade-off between reducing the space required for vertical routing and reducing the amount of horizontal routing since

---

**Figure 1**

**Figure 2**

From the collection of the Computer History Museum (www.computerhistory.org)
in order to put two pins in the same intercomponent gap it may be necessary to move one of the components further away from its best horizontal position and increase the amount of horizontal routing required.

These processes will have reduced the amount of space required for the current set of components as far as possible. If there is some extra space left in the slot an attempt is now made to find the component which will best use that space. When this operation has been completed the slot will usually be very tightly packed with little unused space. However any unused space is now inserted next to the lowest component in the slot in order to increase the width of what will probably turn out to be the next slot. The x coordinates of the components are now fixed allowing the x coordinates of connection targets to be fixed so that horizontal routing can take place. The y coordinates of the components are left undefined and are only fixed as each component has all routes which need to pass beneath it completed. The component can then be allowed to settle on top of the horizontal routing under it. This delayed fixing of y coordinates is necessary because it is extremely difficult to calculate the amount of space required for routing under a component without actually doing it. The next part of the operation is to perform the horizontal routing in the slot. Because all the coordinates have been fixed it is possible to route horizontal segments even though the exact positions of the components are not known. For this routing a system of priorities is used. If there are two or more types entering the slot from below that need to be connected, these are given highest priority so that they are connected as early as possible and do not interfere with the actual intercomponent routing. Next highest priority routes are those which need to pass under components. The effect of completing these is that the components may be placed as low as possible in the slot and take less height. Lowest priority routes are those that only pass from the bottom of the slot to the top, and routes from component sides to the top of the slot. The algorithm operates by performing horizontal routing at increments of one conductor pitch from the bottom of the slot. As many routes and potential routes are completed at each routing level and when no further routing is possible vertical connections are made from the lower ends up to the next level and a fresh attempt is made there. As all routes below each component are completed, the component position is fixed and vertical routing to it is made. In parallel with this activity routes going to the top of the slot for connection in later slots are made and when all the components are fixed and the partial routing completed the processing of the slot is finished. This process continues for all the slots until eventually there will be no more components left to place and all the loose conductor ends will be joined in the final slot. As soon as the layout is finished all the pads are added and an output file is produced containing the details of the layout.

This fully automatic system, when tested on a manually implemented design produced a chip about 1.25 times the linear dimension of the manual design. It was found that there were several instances of awkwardly placed components or densely connected components where the program did not produce an optimum layout. In order to improve the
situation an interactive phase has been added which allows
the program and the user to work together. A menu con-
taining the names of all the components in the design is
displayed on the right hand side of the display screen on
which the design is drawn (see Figure 4). At the beginning
of each slot the user is given the chance to choose the
component positions and orientations. He may either fill the
slot or break off on a partially full slot and instruct the
program to fill the rest of the slot automatically or to leave
the rest of the slot vacant. Using this facility the program
improved on the completely automatic technique giving a
result about 1.15 times the linear dimension of the manual
layout.

It is worth pointing out the reasons the program does not
equal or improve upon the layout produced manually. There
are two main reasons. The main one is that the manual
designer often designs his components to butt together. The
program is unable to detect this fact and will always leave
space for, and perform, some routing between the compo-

Research is under way to test the feasibility of pro-
viding some method to butt together some components au-
tomatically. In a completely "standard cell" system a differ-
ent algorithm is used which takes advantage of all the
aspects of "standard cells". The second reason why the
automatic layout is larger is that designers often generate
special versions of modules to fit into local contexts. This
has the effect of reducing the amount of connection required
had the "standard" version been used. The program, having
no knowledge of what is contained within components, is
unable to adjust component sizes or shapes and must make
do with what is provided. For small numbers of custom
designed IC's the design time/chip size trade-off is a good
one and will become better for even larger numbers of pro-
duction units when the new VLSI processes become com-
monplace and space ceases to be critical.

POST PROCESSORS

The interactive design program produces technology in-
dependent output assuming there are two layers available
for routing. In order to realize the circuit in the required
technology an appropriate post processor is required. Typi-
cally the post processor takes the design file produced by
the layout program and assigns conductor segments to lay-
ers, inserts underpasses where necessary and locates con-
tact windows where interlayer connections are required. It
is at this post processor stage that we need to ask the
question "What is going to happen to the design from
here?" At Edinburgh the design is passed to an independant
suit of drafting programs collectively known as GALEIC.²
The post processor searches the source and library files for
the component definitions and extracts from them the details
of the internal logic specified in GALEIC manual input lan-
guage. This is output to a file together with all the intercon-
nections and contact windows in the GALEIC manual input
language, thus presenting the GALEIC system with a com-
plete design which looks as though it were produced man-
ually. Using special programs in the GALEIC suite it is
possible to automatically generate drive tapes for pattern
generators and other mask-making devices. The use of the
MAGIC suite of programs in conjunction with GALEIC
substantially reduces the time required to design an in-
tegrated circuit. Used properly, the system can reduce design
time by a factor of ten. This is achieved by storing in the
library the complete component specification from the pre-
vious circuits; as the library is built up, all that is necessary
to perform a new design is to select appropriate items from
the library and to specify the interconnections. After this
stage everything can be automatic. The chip is laid out by
MAGIC and the appropriate GALEIC post processor is used
to generate drive tapes for a mask making device. If any
interaction is required, the facilities exist, not only in
MAGIC but also in terms of the substantial graphic editing
facilities in GALEIC.

APPLICATION CONTEXT

The MAGIC suite of programs is applicable to all areas
of IC design in which modularity of design exists. Both the
rectangular components described earlier and standard cells
are laid out, although the latter layout is not as efficient
as if a specialized standard cell program had been used.

The system is most effective when used for irregular de-
signs where patterns are not repeated a larger number of
times such as in memory chip designs. In the latter instance,
the designer can often produce an optimal circuit design in
such a short time that the overhead represented by the extra
space requirement using MAGIC is not warranted. The sys-
tem is applicable to all current IC technologies since suffi-
cient parametrisation exists for the details of individual tech-
nologies to be included in the design specification. The use
of post processors, one for each technology, allows the
design to be optimized for each particular technology auto-
matically.

As VLSI techniques become more commonplace the dif-
ficulty of laying out a chip by hand is going to be so extreme
that unless systems like MAGIC are used, the full potential
of VLSI is not going to be realized.

REFERENCES

1. Schweiker et al., "The LTX System," 1976 Design Automation Confer-
ence.
2. Eades, J., "The Design of an Interactive Computer System for Micro-