R. C. Moore

A computer program has been developed which aids the preparation of artwork for ministick multilayer printed circuit boards. After the logic diagram has been converted into a system wiring diagram, a wiring list is prepared on punched cards along with a description of the ministick assembly frame. The integrated circuit flat packs are identified and located on the frame in a similar manner and a digital computer locates the conductor paths necessary to complete the circuit. This method eliminates much of the time previously required for manual layout of the artwork and makes practical multiple versions of the same circuit layout so as to compare different packaging designs.

COMPUTER-AIDED LAYOUT of *MINISTICK ARTWORK*

The ministick process has been developed at the Applied Physics Laboratory to provide a means by which integrated circuit "flat packs" may be packaged efficiently and reliably.[1] Since this process was to be used almost exclusively for space flight hardware, certain strict requirements had to be fulfilled:

1. Reliability
2. Ease of design
3. Ease of manufacture
4. Repairability
5. Modification ability
6. Form factor (volume density).

To meet the first requirement, the ministick process minimizes the number of connections, providing a rugged assembly frame which makes all connections accessible. This also helps to fulfill the fourth and fifth requirements, while the design of the frame itself satisfies the sixth requirement. Milling and welding techniques provide for ease of manufacture, and a technique using standardized templates aids the ministick layout draftsman during design.

The layout process is still a tedious job, however, and it is vulnerable to the effects of human error. For these reasons it was considered desirable to automate the layout process, using a digital computer to find permissible paths for the necessary conductors and to route them along the appropriate channels on the individual layers of the ministick board. This method would relieve the design draftsman of the responsibility of finding non-intersecting paths for the conductors on each layer, at the same time terminating all the conductors at the appropriate points without error. Final checking with the wiring list or system wiring diagram could be done as the artwork is prepared from the conductor patterns generated by the computer.

To be useful, a computer program designed to automate the ministick layout would have to generate a reasonably good layout in a practical amount of time. As a preliminary design objective, a "reasonably good layout" was defined to be a layout such that the number of layers required was no more than 110% of the number of layers required for a draftsman's layout of the same circuit. Similarly, a "practical amount of time" was defined to be on the order of one to two hours for the layout of a 128-chip ministick board.

Additional requirements included:

1. Variable size boards—up to 128-chip capacity (8 × 16).
2. Variable board format—to allow for different orientations of flat packs, input/output terminations, mounting holes, etc.

[1] C. F. Noyes, "Ministick Packaging—A Further Aid to Satellite Reliability," *APL Technical Digest,* **5**, No. 3, January-February 1966, 2–10.

3. Allowance for different flat pack styles; e.g., 10-pin and 14-pin packages, ¼ inch by ⅛ inch and ¼ inch by ¼ inch packages, and different ways of numbering pins.

Because of these last three requirements, the problem of how to write a computer program to lay out ministick boards was by no means trivial, and a search of the literature pertaining to computer-aided packaging design revealed very little useful information since most of the work reported there was for layout of ordinary two-sided boards or multilayer boards with plated-through holes. Adapting these techniques to ministick, with its high component density and limited board area, while satisfying simultaneously all the above requirements, proved to be impractical.

The solution to the problem lay in a generalized approach to path connections in the plane using a non-Euclidean geometry, thus providing the computer with a means by which it might "recognize" conductor patterns so as to avoid intersections. The problem is one of topology, and a brief digression into the labyrinths of topological maze theory proved edifying.

Mazes and Manhattan Geometry

To a mathematician a maze is just a fixed problem of topology; that is, if the maze were inked on a sheet of rubber, the solution of the maze remains invariant (topologically) no matter how severely the rubber sheet is deformed. The turns and decision points along the correct path from start to finish will always be the same, and will always occur in the same sequence as that path is traversed. It is therefore relatively simple to develop a maze-solving machine capable of solving an arbitrary maze by means of trial and error and capable of remembering that solution.

The ways in which the walls of a maze can be connected fall into two topological classes: "simply" connected and "multiply" connected. Figure 1 illustrates these two topologically distinct types of mazes. A simply connected maze (Fig. 1a) is one which has within it no detached walls. In agreement with its name, this type of maze is simple to solve: merely "walk" through the maze with your right (left) hand always in contact with the right (left) wall. This method, while sure to solve the maze, will not provide the shortest solution, in general. A multiply connected maze is one which has detached walls (Fig. 1b), and is in general considerably more difficult to solve, although there exist algorithms which will solve even the most difficult multiply connected maze.

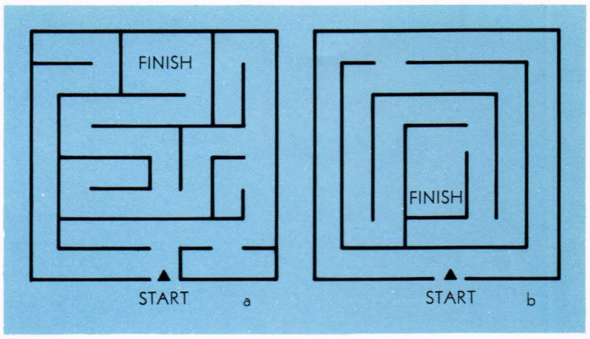


Fig. 1—Simply connected and multiply connected mazes.

It seems logical to ask whether there exists an algorithm which would enable a computer to solve an arbitrary, multiply connected maze such that the path which it discovers is always the shortest path from start to finish. To discuss this question, it is necessary to define precisely the term "shortest path."

If the boundaries of a maze are restricted to be horizontal and vertical straight lines, as is the case in the mazes of Fig. 1, and if the solution is limited to horizontal and vertical lines, then the length of any path from start to finish is simply the sum of the lengths of the horizontal and vertical line segments composing it. Therefore, every maze which has a solution has a shortest path from "start" to "finish." If there exist a number of solutions with the same shortest length, those solutions are considered equivalent, and none is preferred over the others. The maze in Fig. 1b has only one shortest path.

Once "horizontal" or "vertical" has been established, these restrictions imply that, in general, the shortest distance between two points in the plane is not a straight line, and the geometry governing path connections is therefore non-Euclidean. Since distances in this geometry are measured in essentially the same way as distances might be measured by a taxi driver in New York City, this geometry is often called "Manhattan" geometry.

Search and Trace Algorithms

For a computer always to find the shortest path through a maze it must either exhaust all possible solutions, choosing the shortest path, or discover a path in such a way that it can be shown there exist no solutions of smaller length. The search algorithm to be described has been designed so that a shortest path is always the first solution found; if necessary, the method of exhaustion would be used automatically. Once the search algorithm finds the path, a trace algorithm traces back from finish to start, marking the path for future reference.

To find the path as quickly as possible, the search algorithm explores all legitimate passages simultaneously until the goal ("finish") is encountered. As soon as this happens, the trace algorithm begins, so any path longer than the one already found is ignored automatically. The trace algorithm chooses arbitrarily if there are ties for the shortest path.

To describe the algorithms, the maze will be subdivided into square cells which are tessellated to fill the area of the plane enclosed by the maze boundary. The walls of the maze are marked by placing a special symbol in the cells with which they coincide. For example, in Fig. 2 the cells coinciding with the walls of a simple maze have been marked with the letter "X." The start of the maze is marked with an "S" and the finish is marked with an "F."

X	X	X	X	X	X	X	X	X	X	X	X
X			X			X					X
X			X					X	X		X
X	S		X			X	X	X			X
X			X			X					X
X						X		X	X	X	X
X	X	X	X	X		X		F			X
X						X	X	X			X
X	X			X	X	X					X
X			X	X					X		X
X						X	X	X	X		X
X				X							X
X	X	X	X	X	X	X	X	X	X	X	X

Fig. 2—Maze subdivided into cells.

The search algorithm begins at the "S" and assigns a cell mass of 1 to every unoccupied adjacent cell. This concludes the first time step. On the second time step, a mass of 2 is assigned to every unoccupied cell which is adjacent to a cell with a mass of 1. In general, during the Nth time step a mass of N is assigned to every unoccupied cell which is adjacent to a cell with a mass of $N-1$.

At the end of each time step a check is made to determine if the "F" has been encountered. When this occurs, the search algorithm has found the shortest path(s), and the trace algorithm takes over. Figure 3 shows the maze of Fig. 2 after the 24th and final time step of the search algorithm.

X	X	X	X	X	X	X	X	X	X	X	X
X	2	3	X	9	10	X	12	13	14	15	X
X	1	2	X	8	9	10	11	X	X	16	X
X	S	1	X	7	8	X	X	X	18	17	X
X	1	2	X	6	7	X	21	20	19	18	X
X	2	3	4	5	6	X	22	X	X	X	X
X	X	X	X	X	7	X	23	F			X
X	12	11	10	9	8	X	X	X	24		X
X	X	12	11	X	X	X	21	22	23	24	X
X	14	13	X	X	18	19	20	21	X		X
X	15	14	15	16	17	X	X	X	X	24	X
X	16	15	16	X	18	19	20	21	22	23	X
X	X	X	X	X	X	X	X	X	X	X	X

Fig. 3—Maze at completion of search algorithm.

X	X	X	X	X	X	X	X	X	X	X	X
X			X			X	●	●	●	●	X
X			X	●	●	●	●	X	X	●	X
X	S		X	●		X	X	X		●	X
X	●		X	●		X	●	●	●	●	X
X	●	●	●	●		X	●	X	X	X	X
X	X	X	X	X		X	●	F			X
X						X	X	X			X
X	X			X	X	X					X
X			X	X					X		X
X						X	X	X	X		X
X				X							X
X	X	X	X	X	X	X	X	X	X	X	X

Fig. 4—Maze at completion of trace algorithm.

The trace algorithm begins at the "F" and on each time step moves to a cell the mass of which is one less than the mass of the current cell. Ties

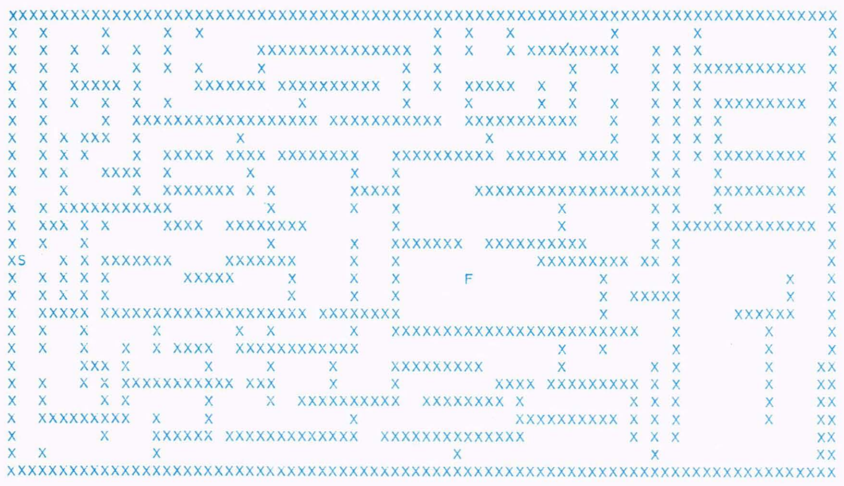


Fig. 5—A multiply connected maze.

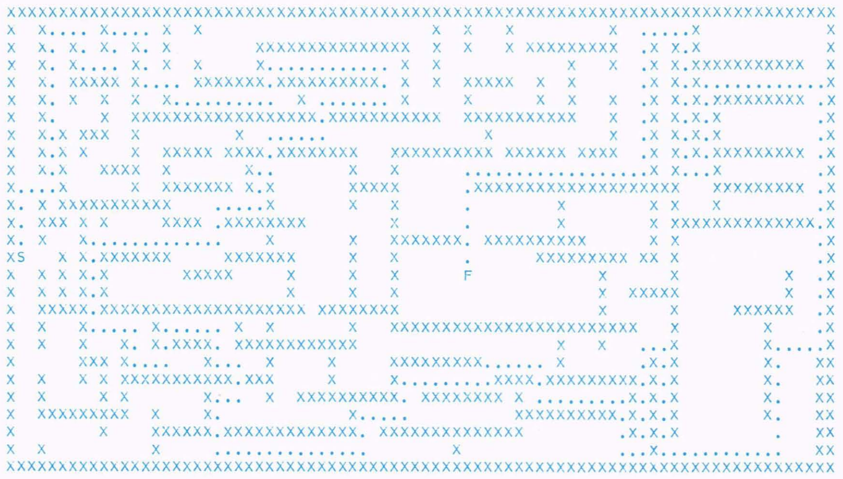


Fig. 6—Computer solution.

are broken arbitrarily, since only one cell can be chosen on each time step. The trace algorithm stores the location of each cell into which it moves, so when the "S" is reached and the trace algorithm terminated, the shortest solution is stored. All the cells with mass numbers can then be cleared, and the path traced is marked with a special character. Figure 4 shows a typical solution where dots have been used to mark the solution path.

If the search algorithm is applied to a maze which has no solution, an N exists such that on the Nth time step there will be no unoccupied cells adjacent to those cells with a mass of N−1. This means that every path terminates in a "dead end," and no solution exists.

Figure 5 is a large multiply connected maze with more than one shortest path. Figure 6 shows the result of applying the search and trace algorithms to this maze. Using a digital computer, Fig. 6 was generated from Fig. 5 in 1.12 seconds (excluding printing time).

Layout Process

The ministick process is designed so that there are no connections between layers. This means that if two integrated circuit leads are to be connected together, the conductor which connects them must lie entirely within one layer of the board. For this reason, large (128-chip) boards may have as many as 20 layers, and small (24-chip) boards will seldom have fewer than ten layers. The layout problem, therefore, can be reduced to the simple problem of connecting pairs of points in a plane (representing one layer of a ministick frame) so that no previously positioned conductor is intersected.

In general, the design draftsman is given a partially filled layer of a board and is asked to connect two points using only the unoccupied channels shown on the ministick artwork template. Since these channels are always horizontal or vertical, they obey the Manhattan geometry, and if the edges of the frame and the conductors already on the layer are considered to be "walls," the problem is that of solving a large, multiply connected Manhattan maze.

A set of points all of which must be connected together by one conductor is called a net. To connect a net of N points, the points may be considered serially and connected as N−1 pairs. Point 1 is connected to Point 2, then Point 2 is connected to Point 3, etc. Using the maze-solving search and trace algorithms, it is therefore relatively simple to automate the layout process, using a digital computer, by treating each net of N points as a set of N−1 mazes which must be solved.

Figure 7 is a basic flowchart of a program to do a ministick layout. It uses magnetic tapes as temporary storage for the nets of points, which are initially read in from cards. The machine has a description of the basic frame (in the form of a planar array of cells marked with the boundaries of the frame edges, mounting holes, etc.) also read in from cards. For each layer, all the nets on tape are examined one at a time. If, during the examination of a net of N points, any one of the N−1 mazes cannot be solved, the entire net is returned to temporary storage to be examined again during the layout of the next layer. When a net has been connected successfully it is traced onto the appropriate cells using a unique conductor number. These cells will be treated as maze walls until the current layer is complete. Once a net is connected it is not returned to storage, so as successive layers are completed there are fewer and fewer nets which remain to be considered.

In Fig. 7, "S" and "F" are the start and finish points for the search algorithm. "Queue" is temporary storage in which the solutions of the mazes are stored until all N−1 mazes of an N-point net have been solved successfully. The last net on tape is a dummy net used to indicate the end of the file, and contains no interconnection information. Switch "A," the program-controlled termination switch, stops the program when the last net on tape is also the first.

Figure 8 is a sample page of the output of a program coded from the flowchart of Fig. 7. Each

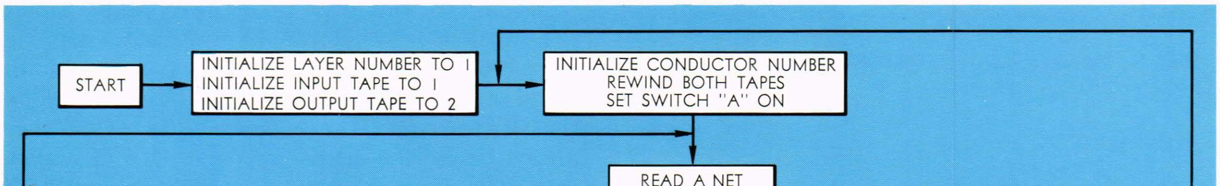
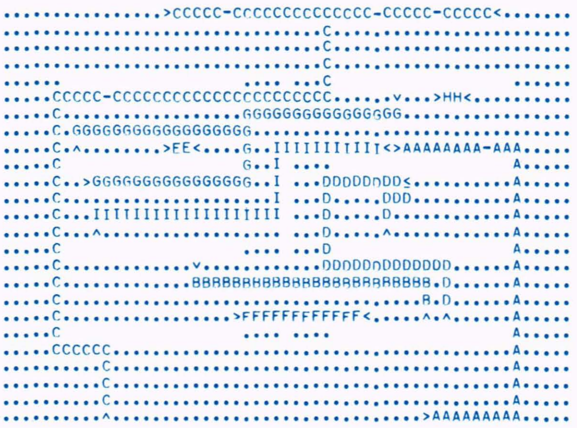


Fig. 7—Basic flowchart of ministick layout program.

conductor is coded with a unique letter of the alphabet. Unused conductor channels are indicated by periods, and the frame boundaries are marked with blanks. A tab (to connect with the integrated circuit flat pack leads) is indicated by one symbol from the set

$$< \quad > \quad \wedge \quad \vee \quad \leq \quad \geq \quad + \quad -$$

depending on the orientation of the channels which connect to that tab, so as to eliminate ambiguities at the termination points of the conductors. The frame used for Fig. 8 has a capacity of 24 flat packs.



```
................>CCCCC-CCCCCCCCCCCCCC-CCCCC-CCCCC<.......
................................C........................
................................C........................
................................C........................
......                  .... ...C                  ......
.....CCCCC-CCCCCCCCCCCCCCCCCCCCCC......v...>HH<..........
.....C..................GGGGGGGGGGGGGGGG.................
.....C.GGGGGGGGGGGGGGGGGG................................
.....C.^........>EE<....G..IIIIIIIIIII<>AAAAAAAA-AAA.....
.....C                  G..I ....                  A.....
.....C..>GGGGGGGGGGGGGGGG..I ...DDDDDDDD≤..........A.....
.....C.....................I ...D.....DDD..........A.....
.....C...IIIIIIIIIIIIIIIIIII ...D.....D............A.....
.....C...^.................. ...D.....^............A.....
.....C                  .... ...D                  A.....
.....C.............v............DDDDDDDDDDDDD......A.....
.....C.............BBBBBBBBBBBBBBBBBBBBBBBB.D......A.....
.....C....................................B.D......A.....
.....C.................>FFFFFFFFFFFF<.....^.^......A.....
.....C                  .... ....                  A.....
.....CCCCCC........................................A.....
..........C........................................A.....
..........C........................................A.....
..........C........................................A.....
..........^...............................>AAAAAAAAA.....
```

Fig. 8—Sample output of ministick layout program.

Results

To consider the quality of the automated layout, a circuit was selected as a standard for comparison of different layouts. The circuit selected consists of two integrated shift counters and uses 23 flat packs. This circuit was an ideal choice because it was small enough to keep the computer run time down to a convenient interval and also because it already had been prepared for ministick packaging by a design draftsman.

Using a frame having four rows of flat packs and six flat packs per row, 14 different layouts of the shift counter circuit were generated. The input data were varied so as to find out the effects on the layout of different methods of pre-processing the data. The computer time required for these layouts, including printing the results, was about 3 minutes per layout.

It was assumed that there exists a particular order of the input nets which will result in a layout with a minimum number of layers. This was verified experimentally. In an attempt to discover what, in general, that order should be, the nets were sorted by size, first in increasing then in decreasing order. In both cases a computer layout was done once for each of two positions of the flat packs on the board. Proponents of the increasing net size argued that if the small nets were put on a layer first, the computer would be able to route the large nets around them and thus get more nets on each layer. The argument for the decreasing sort was that the large nets are more difficult to lay out and should be attempted first.

The following tabulation summarizes the results, including ten layouts in which the nets were entered in a random order:

	Number of Layers		
	10	11	12
Increasing			2
Decreasing		2	
Random	4	4	2

The overall average number of layers required was 11, and the best results were obtained using a random order. The conclusion is that sorting the nets by size does not improve the layout.

Another modification which was proposed was to sort the points of each net so that the points in each row would be considered together. This, it was conjectured, would eliminate excessively long paths and closed loops on a layer. Experiments verified that proper sorting by row could eliminate loops, but the number of layers was not reduced.

Using Manhattan geometry, and always seeking the shortest path, the computer has a tendency always to use the channels at the edge of a stick first. This effect is common among computer path connection algorithms, and is known as the "edge effect." In a ministick layout, the edge effect often blocks off a row of potential tab locations, limiting the number of additional nets which can be placed on that layer. While this problem was not very severe on the 4 by 6 boards, it becomes much more serious on boards with longer sticks, since the edge effect blocks more potential tabs. For example, when eight different layouts of the standard circuit were generated using a frame having three rows of eight flat packs, four layouts required 13 layers while the remaining four required 14 layers. (Using the same frame and chip positions, a design draftsman achieved a layout of this circuit requiring only 11 layers.)

After some extensive modification a minimal edge effect layout program was obtained. The modification involved temporary augmentation of the maze barriers during the preliminary consideration of each pair of points. In all, ten layouts of the standard circuit were generated for the 3 by 8 frame. Seven of these layouts required 13 layers and three required 14 layers. While these results represent, on the average, an improvement over the original program regarding the number of layers, the improvement is much less than had been expected and is insignificant when the time for each layout is considered. The minimal edge effect program required almost 6 minutes per layout, as opposed to 3 to 4 minutes each for the layouts generated without the edge effect modification. It remains to be seen whether edge effect compensation can provide a more significant improvement for layouts on larger frames.

It would seem that there exists some method of positioning the flat packs on the ministick frame so as to minimize the number of layers required for the layout. Since a trial-and-error approach to the problem of finding this method would consume excessive time, a computer program was written to position the flat packs on the frame according to a non-heuristic algorithm.

Positioning and Partitioning

To find the optimal positions for N flat packs would involve examining all N! possible permutations of the chips, assuming there are exactly N positions for the circuits to occupy. For a 23-chip system, assuming each evaluation requires 100 μs, the time required to find the optimum would be

$$(23!)\,(10^{-4}\,\text{sec})\left(\frac{1\ \text{year}}{3.156\times 10^{7}\ \text{sec}}\right)$$

$$\approx 8.191\times 10^{10}\ \text{years},$$

so the method of exhaustion is prohibitive even for a small board.

The technique used to find a reasonable (but not optimal) solution to the positioning problem is based on the assumption that it is best to place flat packs which are strongly interconnected near each other. To describe these interconnections, a nondirected graph is shown with the nodes representing the integrated circuits and the branches representing the interconnections.

Given two nodes, we define the *conjunction* between them to be the number of branches which they have in common. We also define the *disjunction* between two nodes to be the number of different branches which enter either node but not both. Thus the conjunction is a measure of how strongly the two nodes are connected to each other, and the disjunction is a measure of how strongly they connect to the other nodes in the system.

The positioning algorithm begins by placing one flat pack on the frame. It then computes the conjunction and disjunction of that node with every remaining node. The node which has the greatest conjunction with the first node is placed next to it. If there are ties for maximum conjunction, the one with the least disjunction is selected. The two nodes on the frame are then logically *or*ed and are thus treated as one node. The process is then repeated until all nodes are on the frame.

This method is fairly rapid and has yielded good results when used to position computer circuits and cards. It can be used to decide which flat packs should go together on a board in a multi-board system, and is therefore a partitioning algorithm, also.

The positioning program was used to locate the 23 flat packs of the standard circuit on the 4 by 6 frame, and nine different layouts were generated. For comparison, five more layouts were done with the circuits arbitrarily positioned as they appeared on the logic diagram. The following tabulation summarizes the results:

	Number of Layers		
	10	11	12
Arbitrarily Positioned	1	3	1
Computer Positioned	3	3	3

Apparently the number of layers is not a strong function of flat pack position, so it does not seem worth while having the computer position the chips on the frame, since an arbitrary positioning scheme which follows the logic diagram would be much easier to check out and repair after the board is manufactured.

Conclusions and Prospects

By treating the interconnection problem on a ministick layer as a general problem in pattern recognition and by solving the problem as if it were a series of multiply connected mazes, it has been demonstrated that a computer is capable of generating a reasonably good ministick layout in a practical length of time. Pre-sorting of the input data by net size does not improve the layouts generated; in fact, the best results are obtained when the interconnection data cards are read in in a random order. Pre-sorting of each net can be helpful in eliminating closed loops in a conductor on a layer, but careful positioning of the flat packs on the frame provides no significant decrease in the number of layers.

While the edge effect does not appear to be a major problem on 24-chip boards, precautions may have to be taken when larger boards are considered. Perhaps other simple modifications can be made to improve the layouts generated; these may become obvious when larger boards are considered.

To be really useful, the computer should be capable of generating the ministick artwork itself, since this is the most time-consuming part of the ministick layout procedure. With a program to generate a suitable output tape from the results of this layout program, an automated drafting table could be used to generate the artwork directly, provided the accuracy were high enough not to prohibit the photo-reduction of the artwork to actual size. Automatic tables such as this are made by the California Computer Products Company and the Gerber Scientific Instrument Company.

In the future, when part or all of the logic design itself can be automated, it is not inconceivable that the computer will be able to design a circuit from preliminary specifications, referring to a library of available (and acceptable) integrated circuits, generate the interconnection pattern and wiring list, partition and position the flat packs, and generate the ministick artwork, all with a minimum of human intervention.

Excerpts from the REPORT of the DIRECTOR *of the Applied Physics Laboratory*

JULY 1, 1965—JUNE 30, 1966

To the President of the University:

Technical Activities

The technical operations of the Laboratory have generally continued in the pattern of the past two years. The effort is concentrated in four principal areas: (1) Navy Surface Anti-Air Missile Systems, (2) Fleet Ballistic Missile Systems, (3) Space Development Program, and (4) Supporting Research and Exploratory Development.

Recently an analysis was made of the distribution of effort in the Laboratory among different types of technical activity. The results were quite revealing of the diverse nature of the Laboratory's operations. The types of activity, percent of each, and examples of the work involved are given below:

1. Research 13%

This includes basic independent scientific research carried on largely by the APL Research Center and applied research, such as the investigation of clear air turbulence for the Air Force.

2. Exploratory and Advanced Development 17%

This includes most of the work of the Aeronautics Division in the area of jet propulsion and such developments as the gravity-gradient satellite stabilization system.

3. Engineering 29%

The bulk of the in-house engineering done by the Laboratory is the design and fabrication of satellites. This category also includes technical direction and technical support of engineering performed by contractors in guided missiles and radar.

4. System Engineering and Integration 19%

The major portion of this work is done in the surface missile program. It includes formulation of system requirements, system analysis and simulation, and evaluation of prototype subsystems in a system environment.

5. Technical Evaluation of Operational Systems 22%

This category relates to analysis and test of operational systems to determine their performance, reliability, readiness, or other operational capability. The Laboratory's effort in the Polaris program falls into this category, as well as certain recent simulation work in connection with tactical air operations.

Surface Anti-Air Missile Systems—The Navy's commissioned guided-missile fleet of 12 cruisers, 26 frigates, 23 destroyers, and 3 carriers is armed with Talos, Terrier, and Tartar missiles, which were conceived by the Laboratory and developed in cooperation with its associated contractors. In 1962 the Laboratory was asked by the Navy to take technical responsibility for the shipboard fire-control systems which were initially developed for