P. W. Case
H. H. Graff
L. E. Griffith
A. R. LeClercq
W. B. Murley
T. M. Spence

Solid Logic Design Automation

Abstract: This paper describes the unique features of a set of IBM 7090 programs which provide design assistance to engineers who use Solid Logic Technology. These programs were applied in the design of the IBM System/360.

Introduction

Automatic design aids have become necessary for the efficient use of engineering manpower in developing modern digital systems. The design of such systems involves the handling and documentation of vast amounts of information. Without design aids, the engineer is required not only to perform his vital task of specifying the system's internal logical structure, but also to perform manually all the tedious chores of detailing, recording and checking his design.

The concept of design automation for digital systems has been previously discussed.[1] Operating within this concept, an engineer must state in rough-draft form the arrangement of logical elements for the system he is designing. Once he has done this, computer processing can assist with each further stage of design detailing, and furnish up-to-date documentation of the state of the design. The basis for this process is a central file kept on magnetic tape, which serves as the prime definition of the design in much the same way that master drawings formerly served. Automatic printout of logic diagrams by the computer replaces the outmoded process of obtaining blueprints.

Design automation for the System/360

• *Logic design*

The design of the IBM System/360 required that new design automation techniques be provided. Since the system philosophy[2] demanded high performance objectives and strict compatibility, with a consequent increase in the volume and complexity of logical design, a need for more efficient design-information handling and documentation became evident.

Thus, previous concepts for retaining accumulated designs and producing computer-printed logic pages were included in the design procedures, but novel features were added to reduce the burden on the engineer. These features increase the computer's ability to use data on the central design tape to produce a variety of documents. This increased ability eliminates many of the errors which formerly were caused by manual generation of data.

A set of simulation programs was developed so that System/360 designers might evaluate their logic designs before they construct hardware models. Several features of these programs provide greater flexibility of operation than previous simulation programs have been able to attain.

• *Packaging*

An early systems planning decision was that the System/360 would use a new circuit-packaging technique called Solid Logic Technology (SLT). This decision provided the primary motivation for developing the design automation techniques described in this paper.

As described by Davis, et al.,[3] SLT microelectronic circuits are encapsulated into modules. From 6 to 24 of these modules are mounted on "small cards" of various standard sizes. Etched wiring on both sides of a small card interconnects module terminals with female connectors at the base of the card. The small cards, in turn, are mounted on larger wiring boards whose various terminal pins make pluggable contact with the small card con-

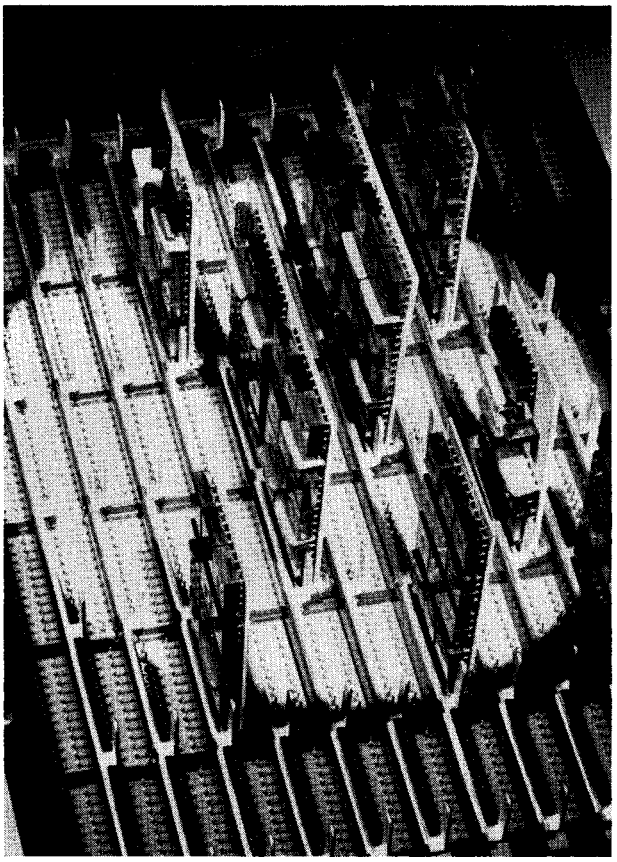


Figure 1 **Small cards mounted on large etched-wiring board.**

nectors. The "large boards" also have etched wiring signal paths on both sides.

Interconnection between large boards is accomplished by flat, flexible cables.[4] Figure 1 shows small cards mounted on a large board. Predesigned small cards with circuit modules attached are the smallest units of hardware with which a System/360 designer works.

New programs were developed (1) to assign each block appearing on the logic diagrams to appropriate small-card units, (2) to assign labels to small-card connection points associated with each input and output line from the blocks on logic diagrams, (3) to assign each small card to a position on a large board, and (4) to compute the routing, sequencing, folds, and lengths of the flat cables that interconnect large boards.

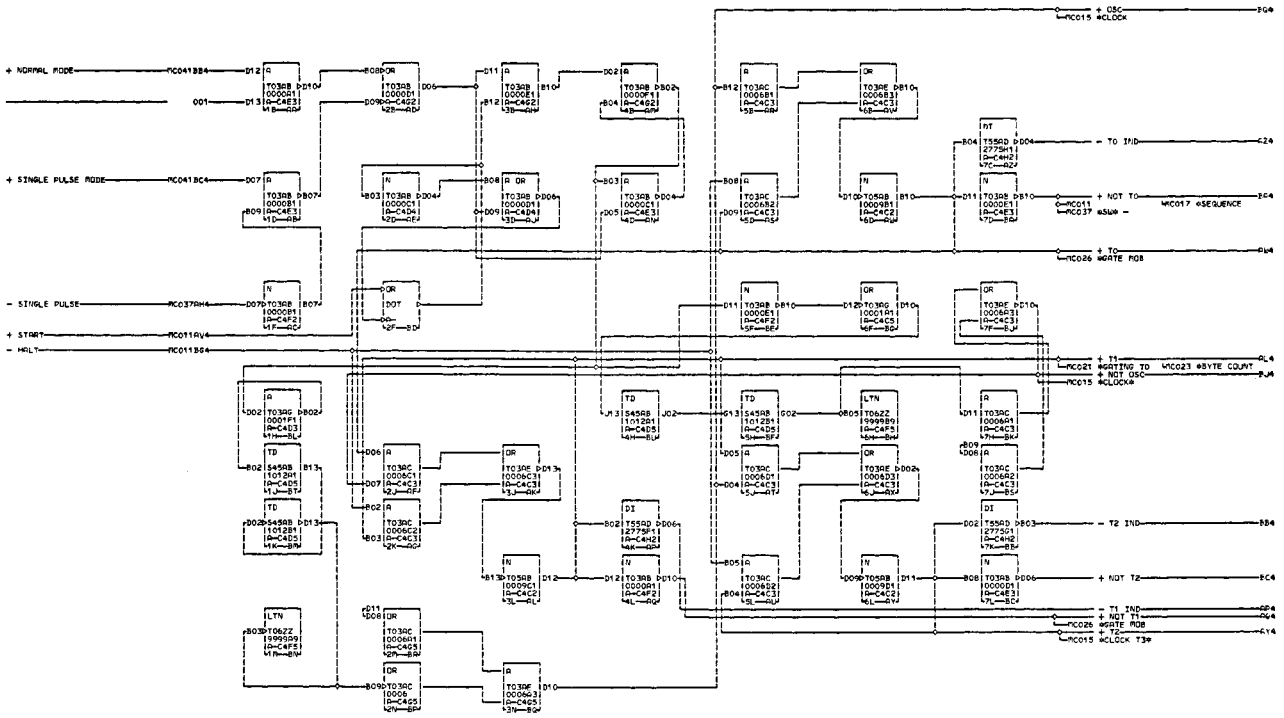
Other new programs were created to automatically determine the layout of etched wiring patterns on the large boards.

Logic Design Accumulation Process

The Logic Design Accumulation Process provides an up-to-date record of the machine design status at every stage of development. The primary document of this record is a computer-printed logic diagram like that shown in Fig. 2.

The process begins when an engineer draws a diagram

Figure 2 **Sample computer-printed logic page.**



showing how a group of logical elements (e.g., AND or OR blocks) should be connected to perform a desired machine function. The engineer makes this drawing on a special form which enables a keypunch operator to transcribe his design onto punched cards. The punched-card data are then processed through an IBM 7090 computer and stored on magnetic tape in the Design Automation Logic Master File. The computer also produces at this time a printed diagram of the logic specified by the engineer. Whenever the engineer makes a change in his design, the Logic Master File is revised, and an updated logic page is printed. As Fig. 3 shows, other programs will subsequently obtain data directly from the Logic Master File.

Throughout the design process, the engineer is provided with a complete set of the most recent computer-printed logic pages specifying the system design. These pages are numbered and titled to identify the logic function that is depicted. The names and numbers of all signal lines entering and leaving the page are shown. Each logic block on the page has space provided for printing (1) the type of logical function performed by the block (AND, OR, INVERT, TRIGGER, etc.), (2) the identification number of the microelectronic circuit represented by the block, (3) a number specifying the type of small card on which the circuit appears, (4) a designation representing the portion of the small card on which the circuit appears, (5) a number stating where the small card will be located on the large board, and (6) identification of the small card terminals which make contact with pins on the large board. Figure 4 shows the information associated with a sample block from a logic diagram.

Although all the information just stated must eventually appear on the logic page, the engineer does not have to specify it all in his initial input to the Logic Master File. He is encouraged only to specify initially the signal-line names and call out the logic function and microelectronic circuit numbers associated with each block on the diagram; subsequent programs will assist him with further detailing. Logic at this early stage is said to be "implemented" because its operation is completely specified even though it is not yet packaged or converted to hardware. Checking programs are used after the logic is implemented to test the design on the Logic Master File against a set of circuit interconnection rules. These programs will discover, for example, whether there are any overloaded circuits.

The purpose of the logic diagrams, as already stated, is to provide designers with an up-to-date picture of the system's design status. This statement implies that design changes are expected, and can be automatically recorded on the logic sheets. The new programs developed for the System/360 simplify the problem of recording changes. One of these programs provides automatic cross-referencing of changes. Figure 5 gives an example of this. Figure 5a

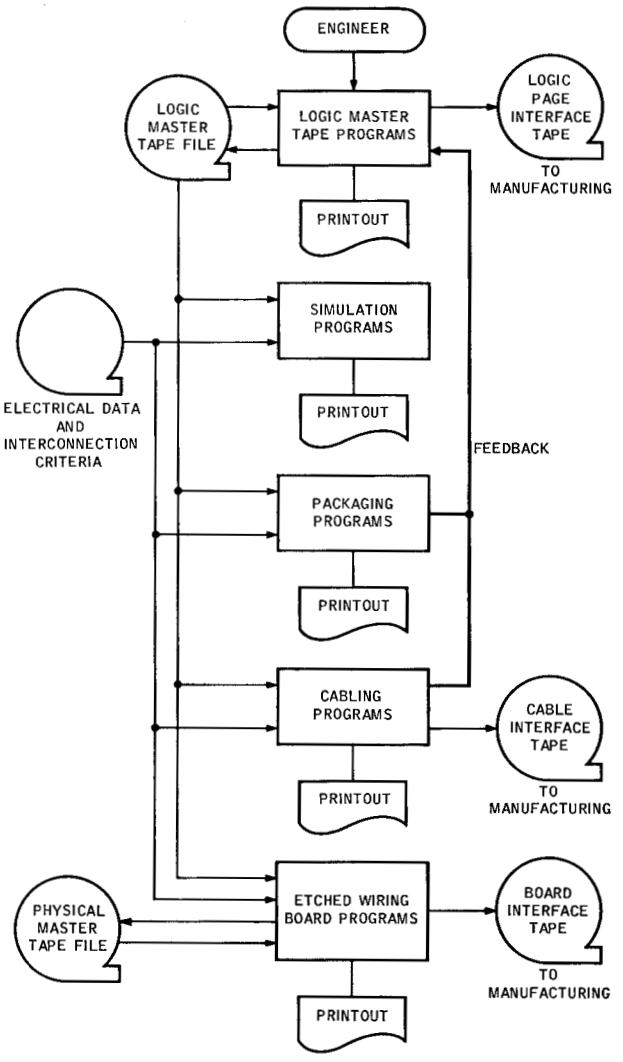


Figure 3 **Solid Logic Design Automation program sets.**

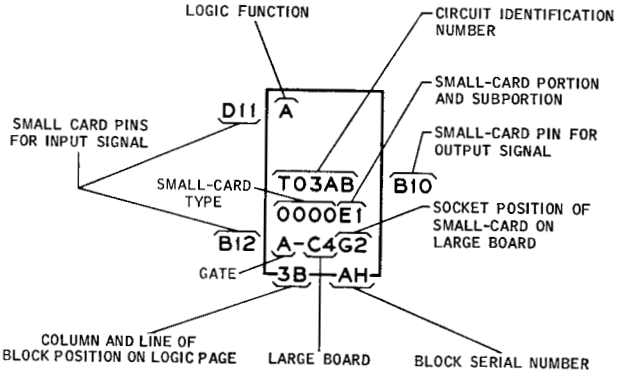


Figure 4 **Information associated with a sample block on logic page.**

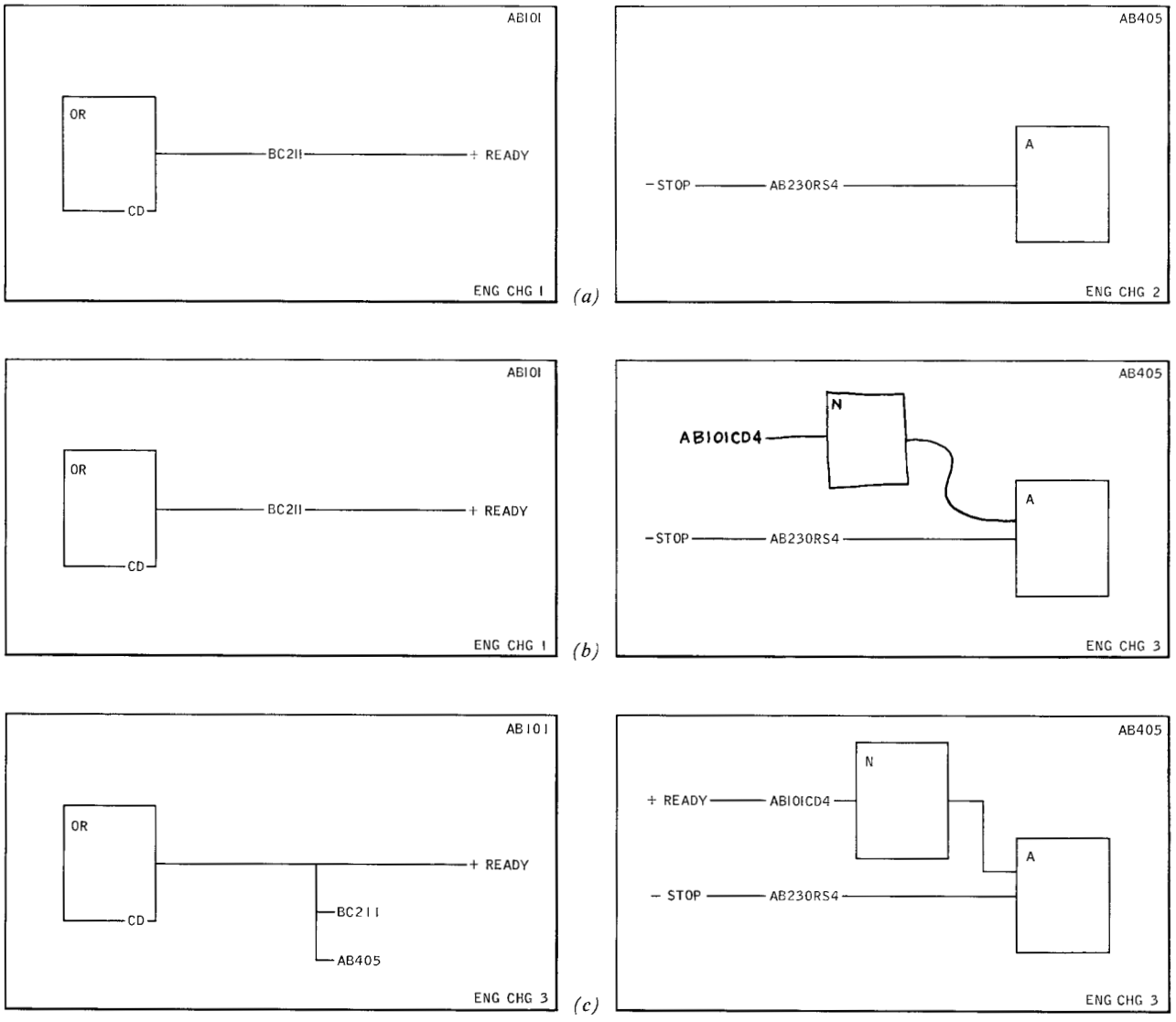


Figure 5 **Example illustrating automatic cross referencing of design changes.**
a) Pages prior to change;
b) Changed page submitted by engineer;
c) Pages produced by programs.

shows portions of two logic pages (pages AB101 and AB405) before changes. Suppose the engineer wishes to use a signal generated on page AB101 as input to a block on page AB405. He then marks up page AB405 as shown in Fig. 5b: he specifies the incoming signal line by writing the page (AB101) and location within that page (block CD, line 4) where the signal was generated.

Using the signal line code (AB101CD4), the computer can search the Logic Master File to get the name of the signal (+READY) and print it on page AB405. It will also automatically revise and print the new usage of the signal on page AB101. Figure 5c shows the appearance of the pages after they have been revised. This unique program allows automatic cross-referencing of signal usage and immediate updating of all pages affected by an engineering change. It reduces the amount of information to be supplied manually, and helps to avoid redundant and ambiguous entries of signal lines into the design.

Permanent assignment of eight-character signal-line codes (called "net numbers") has a further benefit in the latter stages of design. That is, checking of wiring and cabling lists will be facilitated since all net numbers in the lists refer directly to logic pages on which they originate.

Another novelty of the Solid Logic Design Automation programs is called Version Design Processing. Once the basic design of a system has been completed, the design

versions resulting from the addition of various optional features are recorded as modifications to the basic system. Because the unaltered part of the basic design does not need to be duplicated, redundancies and ambiguities are avoided. Composite logic pages are automatically produced for each version which show the appropriate feature superimposed on the basic logic. One of the most significant advantages of Version Design Processing is that the programs automatically produce altered feature pages when the basic design is changed.

Logic simulation

The logic design of the System/360 was made more efficient through the use of logic simulation programs. These programs were devised to enable engineers to predict the performance of their proposed designs before they build hardware models of them. Several new features were incorporated into the operation of this simulator: (1) the logic to be tested is obtained directly from the Logic Master File by the simulation program, (2) the simulator can account for nominal transit and switching delays, (3) it can handle logic feedback loops, (4) it examines the simulated logic only at the times when some element has changed state, and (5) at examining time it checks only those elements which could possibly change state in the future as a result of the present change of state.

To use the simulator an engineer specifies what portion of logic he wishes to test. Since the logic is already on the Master File, the tape serves as the input to the simulator; this eliminates the need for an engineer to manually generate the logic for use in the simulator. Other data that the engineer must supply are a time scale for the simulation run, and the specific times along this scale at which he wants input nets to switch logical states.

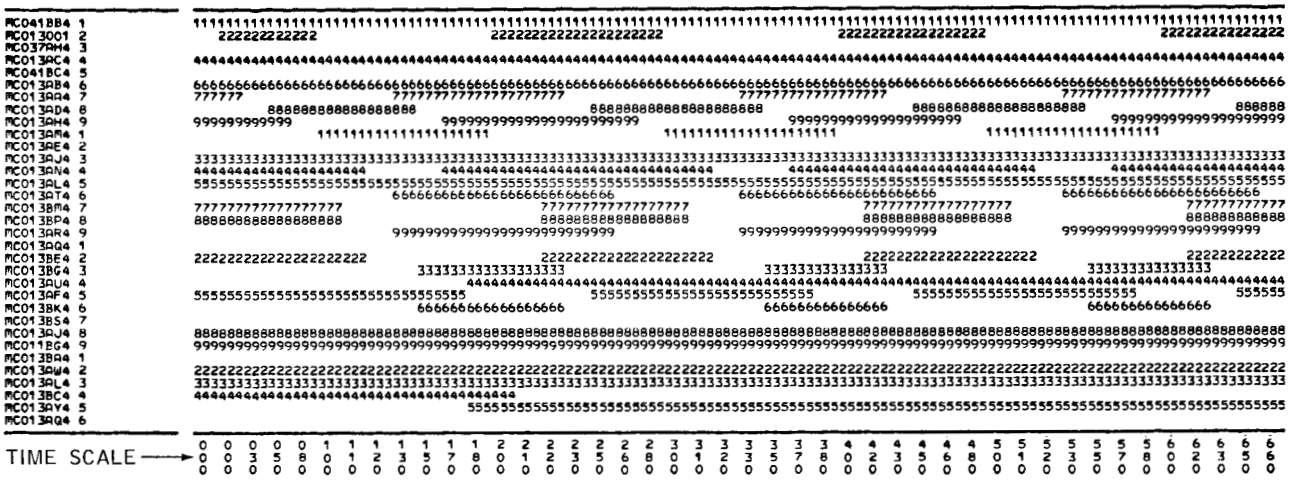
When the coded logic from the Master File enters the simulator it is combined with data from another magnetic tape which specify the delays of each logic block. This combination of data is translated into a set of coded equations that defines the logical state of each net as a function of the delays and the logical states of other nets. The functional equations are then entered into a section of the simulator's core storage called the Functional Table.

Another section of core storage contains a Forward-Referencing Table. For each net in the logic, a list is stored to identify all the other nets whose logical state depends on the state of the given net. Whenever a net changes state, the simulator refers to its entry in the table. It determines which nets may subsequently be affected by that action. By doing so, it eliminates the need to interrogate every net in the logic under test, thus reducing running time. Also, this feature makes it possible for the simulator to consider feedback loops in the logic circuits.

A third section of storage is occupied by a Switching Events Table. The switching of a signal at a designated time is called an "event." Events are listed chronologically in the table along the time scale specified by the engineer. Initially, the events associated with the input nets are entered at the appropriate times in the table. For example, the engineer may have specified that input net x switches "on" at time-unit 1 and switches "off" at time-unit 12. Thus, events are entered in the table at times 1 and 12.

However, the Forward-Referencing Table might indicate that an event on net x could affect the logical state of net y. The entry in the Equation Table for net y may then state that net y turns "off" 4 time units after net x turns "on." Therefore, when the event of x switching "on" is executed at time 1, it will generate a new entry in the Events Table. This new entry will state that net y switches off at time 5.

Figure 6 **Sample sequence chart obtained from logic simulator.**



Since time-unit 5 is the next event in the table, the simulator next determines what effect this event will have in producing future events, and enters any future events at the appropriate place in the table. In this way the Events Table is continually revised throughout the simulation run.

The use of the Forward-Referencing and Events Tables is a valuable utilization of storage space since, normally, only about 1% of the logic elements under test are active at any time. Thus, total running time is conserved by the simulator's ability (1) to examine logic only at those points on the time scale where it encounters an event, and (2) to examine only those logic elements affected by an event.

The results of the simulation run are provided to the engineer in the form of either a timing chart or a sequence chart. To produce a timing chart the simulator samples and prints the status of every net in the logic at equal time intervals. To produce a sequence chart, it prints only at the times corresponding to an event.

Figure 6 shows a sequence chart for the logic page given in Fig. 2. The left part of the chart is a list of all the nets on this logic page for which charting was requested. Three identical columns are printed after one or more of the charted nets has changed state. There will be an entry in a given row of a column if the corresponding net is "on" at time of printing. The time scale along the base of the chart designates the simulated times at which the event occurred.

Simulator timing charts, which can show the activity of up to 100 nets, may be compared to the output that would be observed on a multitrace oscilloscope if a probe were attached to each net. By studying such charts, engineers are often able to spot undesirable timing situations and correct their design before any hardware models are built.

The simulator just described is able to approximate the performance of 3000 to 4000 logical elements through 10 to 20 clock cycles in less than 30 minutes of IBM 7090 computer time.

Packaging and cabling programs

Once the design of the logic is satisfactory, the next problem is to specify the physical layout of logic as it will appear in the machine. As discussed earlier, the System/360 logic is packaged using SLT modular circuitry. New design automation programs have been developed to assign logic elements to standard small cards, to assign small-card pin labels to appropriate logic block terminals, to assign the small cards to positions on large wiring boards, and to determine cabling requirements.

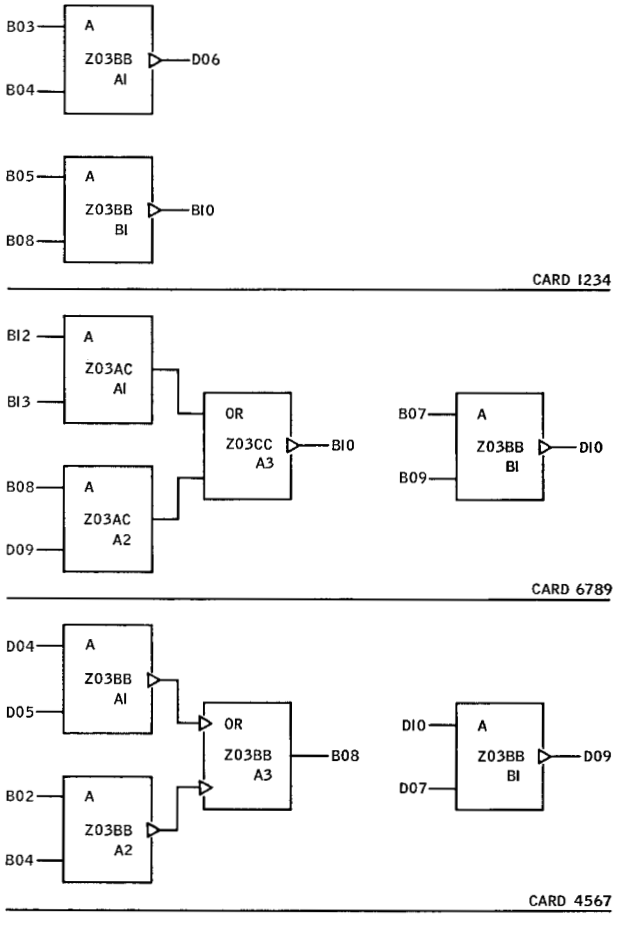
- *Assignment of logic to cards and boards*

1. Formation of logic groups

The Partitioning Program examines a section of logic contained on the Master File and divides it into groups of blocks for assignment to small cards. Each logical element on the tape is represented by a number (e.g., Z03BB) which specifies a standard IBM circuit. This number implies the logical function performed by the element and the electrical characteristics of the circuit.

The total library of small-card packaging units consists of many cards. For purposes of example suppose, however, that the library contains only the three cards shown in Fig. 7. This figure indicates that card 1234 has two Z03BB circuits contained in the modules mounted on it. One circuit is in portion A of the card and the other is in portion B. All input and output terminals of the circuits are connected via pins to the large-board signal paths. Card 6789 has four circuits contained in its modules. Circuits Z03AC are connected by wiring paths on the small card to circuit Z03CC. These three circuits are in portion A of the small card and are in subportions 1, 2, and 3, respectively. The circuit terminals that are connected internally on the card are not available to the signal paths on the large board. Circuit Z03BB is connected on the card independent of the other circuits.

Figure 7 **Library of small cards.**



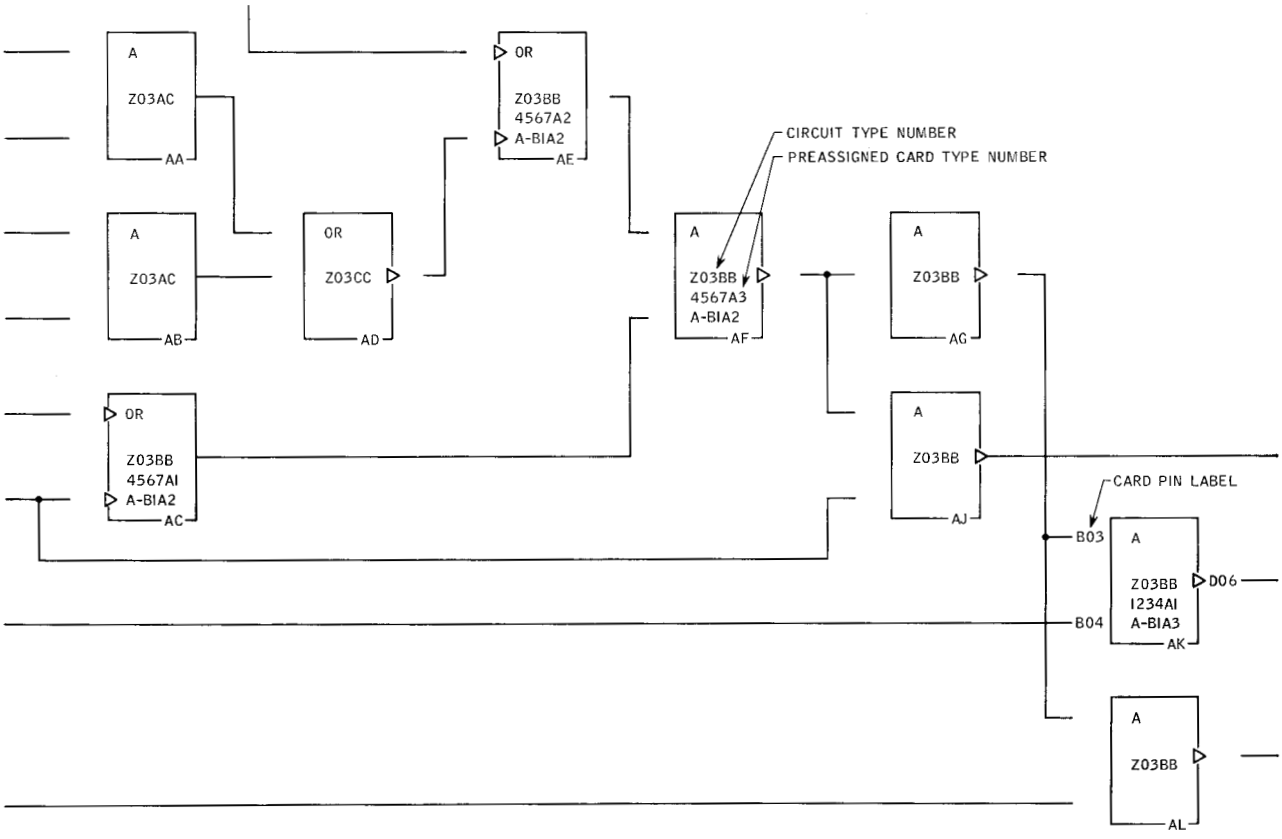


Figure 8 **Appearance of logic diagram before partitioning.**

The circuit configurations encountered on the previous two cards are designated as "non-functional" portions. This is because they are either unit groups or are clusters which *always* appear together on a single small card.

The third card illustrated (card 4567) has three Z03BB circuits interconnected into portion A, subportions 1, 2, and 3. Since Z03BB can also appear as a unit circuit as shown on the previous cards, the collection labeled portion A is called a "functional" portion. Portion B of card 4567 is again an independent circuit.

It is important to remember that the small cards with mounted modules are available as prepackaged units; the computer's function is to assign the blocks on logic diagrams to appropriate portions and subportions of appropriate standard card units.

Now, suppose that an engineer wishes to have the logic shown in Fig. 8 assigned to small cards. At the stage of design represented by this figure, the engineer has manually assigned block AK to card 1234, portion A, subportion 1; he has manually assigned blocks AC, AE, and AF to card 4567, portion A, subportions 1, 2, and 3; all other blocks are to be automatically assigned. (Refer to Fig. 4 for a review of the information contained on a logic diagram.) The engineer's manual assignment of block AK was prompted by his knowledge of conditions such as heating or electrical interference which caused him to have a preference for the particular assignment chosen.

The engineer's manual assignment of blocks AC, AE, and AF was prompted by his preference for the use of a functional portion in the packaging. The partitioning programs would otherwise choose nonfunctional portions for the packaging of these circuits.

The Partitioning Program extracts the logic from the Master File and checks each logic block against a set of partitioning criteria obtained from another tape. The criteria are used to determine groups. The groups of blocks formed by the program for this example and the reasons for each group formation are listed in Table 1 (page 134).

2. *Assignment of groups to cards*

The next step of partitioning is to assign each previously unassigned group to a small card using a process described by Haspel.[5] First, the computer program chooses from the card library a minimum set of cards which is capable of packaging all the groups.

Then the computer goes through a routine which assigns to the same card groups that are "close" together. To do this, the program arbitrarily selects one of the

Table 1 **Formation of logic groups.**

Group number	*Logic blocks comprising the group*	*Group name*	*Reasons for the group formation*
1	AG	Z03BB	This block is not yet assigned to a card. Partitioning criteria require Z03BB to be a unit group which exists as a separate portion on a small card.
2	AJ	Z03BB	Same as Group 1.
3	AL	Z03BB	Same as Group 1.
4	AD,AA,AB	Z03CZ	The criteria require these three blocks to *always* be together on a small card. They are therefore treated as a group.
5	AK	blank	This block is not a candidate for partitioning, since it has been preassigned by the engineer.
6	AF, AE, AC	blank	These blocks are not candidates for partitioning; they have been preassigned by the engineer into a functional portion. If called upon, partitioning would have treated these blocks as three distinct groups, each similar to Group 1.

cards. A "portion" of this card is selected for consideration and a group is assigned to it. (Assume for simplicity that no groups have been preassigned.) Next, another portion is selected and the "best candidate group" is chosen for assignment. A "candidate group" is one whose configuration will fit the configuration of the portion. The "best candidate group" will be the candidate group that is "closest" to the already assigned group. The "closest" group is that one which shares the most nets with the assigned group(s) and has the least number of nets not shared with the assigned group(s). If more than one of the candidates meet this criterion, an arbitrary choice is made.

Now, with two groups assigned to portions, the computer selects another portion and finds the candidate group that is closest to the two assigned groups. In this manner, each card in the set selected from the library is examined portion by portion until all groups have been assigned. Of course, in some cases, there will be more portions available than there are groups to fill them, and the program must decide which portions will not receive an assignment.

For the example logic shown in Fig. 8 the groups could be assigned as follows by the process just described:

Group 1: Card type 4567, portion B, subportion 1
Group 2: Card type 6789, portion B, subportion 1
Group 3: Card type 1234, portion B, subportion 1
Group 4: Card type 6789, portion A, subportions 1, 2, 3
Group 5: Preassigned
Group 6: Preassigned

3. Assignment of cards to boards

After groups have been assigned to cards the next step of the partitioning process is to assign the cards to appropriate large boards. This process is much the same as the group assignment process in that cards which are close to each other are assigned to the same board. Here, closeness of cards has exactly the same meaning as closeness of groups; the closest candidate card is that one which shares the most nets with the already assigned cards, and has the least number of nets not shared with assigned cards.

4. Assignment of pin labels

When the partitioning process has been completed, enough data are available on the Master File so that labels associating small-card connection pins with the signal lines for all blocks on the logic diagrams can be assigned. Pin label assignments are also entered into the Master File.

The diagram of Fig. 9 shows how the logic of Fig. 8 will appear after all assignments have been made. This diagram indicates that each logic block has been assigned to a portion and subportion of a small card, that terminal-pin labels have been assigned, that small cards have been assigned to large boards, and that cards have been assigned to socket positions on the large boards. This last problem of assignment has yet to be discussed.

• *Assignment of small cards to large-board socket positions*

Interconnections between small cards are accomplished by the etched wiring paths on a large board. Because wiring the board involves problems of signal-path routing and requires that intercard signal connections be as short as possible for good circuit performance, it is desirable to make the best possible use of the flexibility that is available for positioning the small cards on a board. The Placement Programs compute positions for a set of cards.

Two basic placement programs are provided for the engineer, an Algorithmic Program and an Interchange Program. They may be used individually or successively. Since the engineer himself will sometimes wish to specify the exact placement of some of the small cards on the large assembly, both programs make provision for this situation. The programs give priority to the predesignations and will assign the remaining small cards to the positions available.

1. Algorithmic Program

To make assignments using the algorithmic method, the program first selects a board from the already chosen set of boards. It then compiles a list of all cards to be positioned on that board and another list of all available socket positions on that board. (A socket is available if it has received no prior assignment, or has not been excluded as available by the engineer.)

Assume, for example, that one card has been assigned to a socket by the engineer. This card and socket do not appear in the tables. The program now picks card C_i from the table and tries it, in turn, in each socket S_j. Each time the computer tries a card-to-socket combination C_iS_j, it notes the location of all pins which must be interconnected between the already assigned socket(s) and the trial socket. It then computes the half-perimeter of the smallest rectangle which can enclose the interconnecting pins of a net that is common to both sockets. If there are more than one common nets (and there usually are), the computer calculates the half-perimeters of the rectangles enclosing each net, and adds them together.

This sum N_{ij} is determined for all possible card-to-socket combinations C_iS_j. From all N_{ij}, a particular N_{pq} is found such that

$$N_{pq} = \max_i [\min_j (N_{ij})].$$

Using this criterion, card C_p is assigned to socket S_q; the card and socket so assigned are removed from the tables, and the process is repeated until all cards have been assigned. In practice, the Algorithmic Program has assigned 60 small cards to socket positions on a board in 10 minutes of 7090 computer time.

2. Interchange Program

When all cards have been given assignments to sockets, the Interchange Program attempts to improve the "wireability" of the board. The wireability factor equals the sum of all N_{ij}, where each N_{ij} is computed as before. The lower the factor, the more wireable the board.

In attempting to improve wireability, the computer tentatively interchanges each card with every other card on the board. At every tentative interchange, $\sum N_{ij}$ is computed. If a lower sum than that produced by the preceding arrangement is encountered, a permanent reassignment of cards is made. The interchange process continues until no interchange that will produce a lower

Figure 9 **Partitioning placement, and pin assignment information added to logic diagram.**

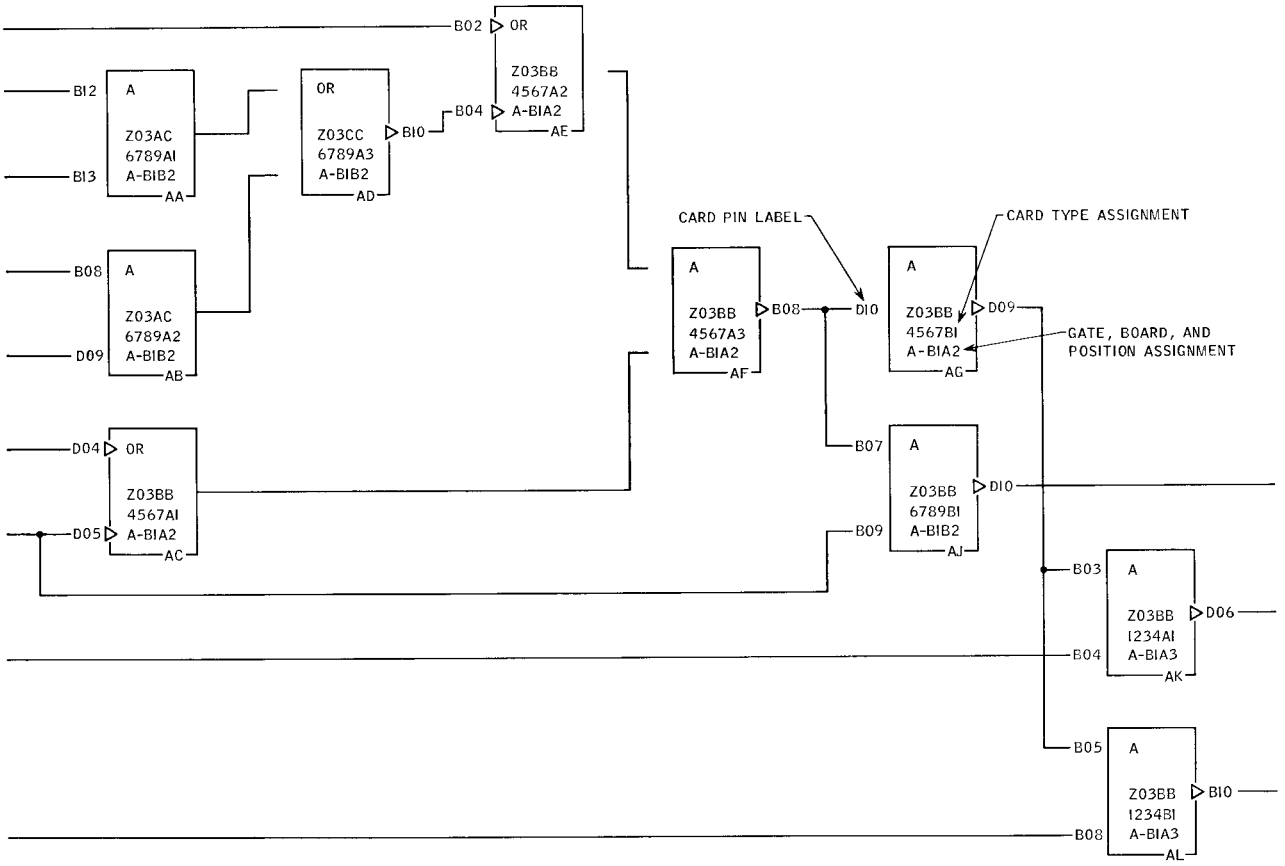


Figure 10 **Cable block diagram.**

$\sum N_{ij}$ is possible. The engineer uses the wireability factor as a means of comparing the efficacy of various placements. In practice, the Interchange Program takes from 15 to 40 minutes of 7090 time to evaluate a 60-card board for wireability and make reassignments.

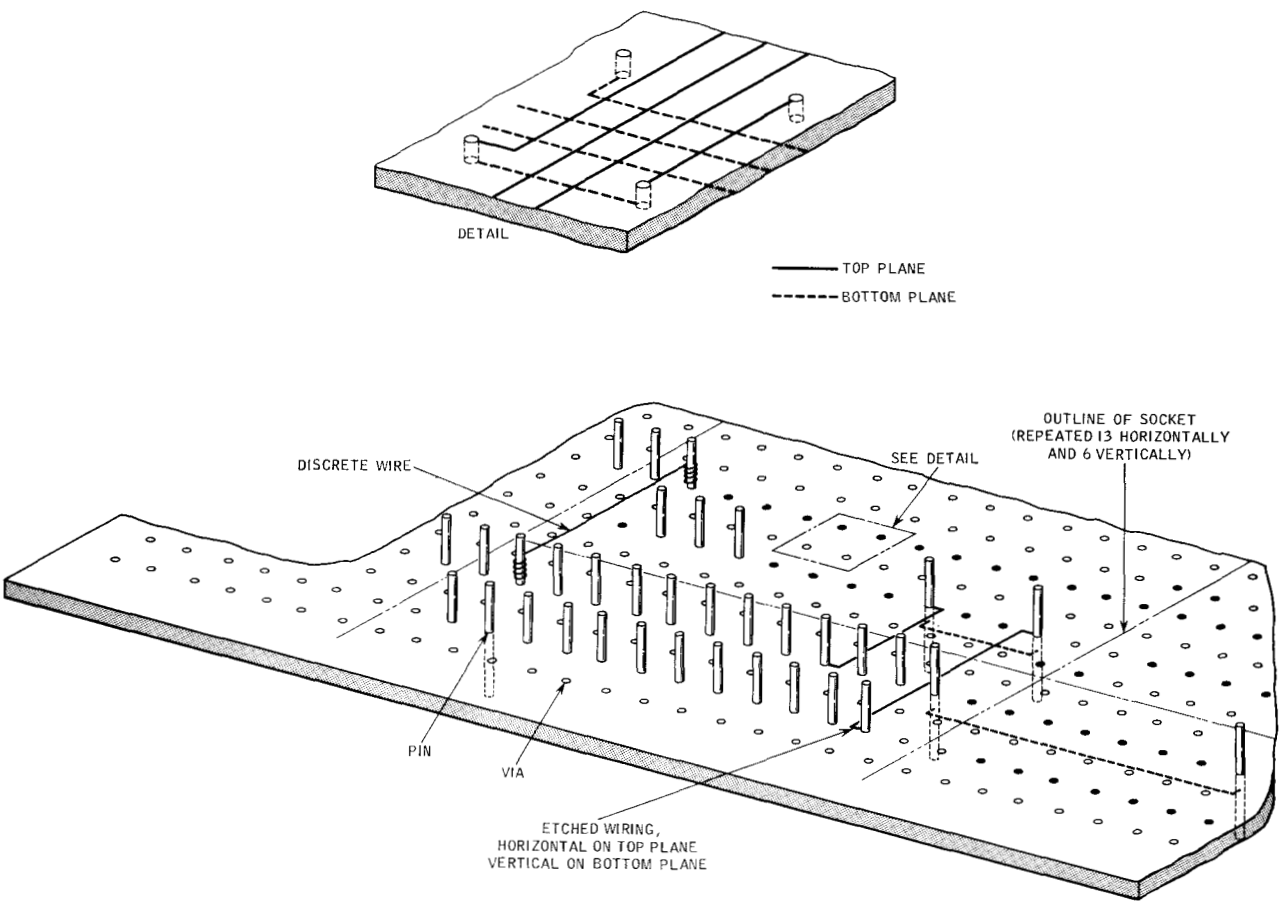
• *Cabling Programs*

Use of flat cables poses new challenges to the designer in obtaining an efficient and accurate physical (mechanical) layout. Highly accurate information specifying the length and position of folds is required. In addition, sequencing and routing must be such that the total thickness of several cables does not exceed the cross-sectional depth of the cable channel, and such that cables emerging from a channel at a given point are in the same relative sequence as the sockets to which they connect.

The Cabling Programs will, given the endpoints of the cables desired, compute the routing, sequencing, folds, and lengths of flat cables. To do this, an additional input is provided by the engineer, who draws "cable block" diagrams as shown in Fig. 10. These diagrams contain pairs of blocks which identify each cable and the large-board sockets it uses.

The Cabling Program first considers cables which are plugged into sockets nearest the routing channels, and places them into a table of "available" cables. From this group, those cables which have only one routing path are selected. An examination of the paths of these cables, along with the matching of folds as a cable's direction changes, determines an initial placement in the routing channels. As each cable is placed, other sockets are made accessible and new cables are added to the list of availables. As each cable is placed in its path along the channels, a summary of build-up at channel intersections is maintained. This is used as a factor in choosing a route for a cable when there are alternate paths which are equally desirable. During the routing of a cable, lengths are subtotaled for fold marking. Each subtotal is used to

Figure 11 **Identification of etched-wiring board features.**



indicate to the cable manufacturing process the midpoint of a mark that shows where it is necessary to fold a cable when a change of direction is indicated or where it is necessary to clamp a cable (or cable set) for additional support.

Upon completion of the design of a set of cables, magnetic tapes are produced for use in the manufacturing process. The primary advantages accruing from the use of these programs is accuracy of design data and reduction in manual effort and cost.

Although the features provided for design assistance described above are important, the most essential design aid provided to the engineer using the SLT technology is the preparation of etched board wiring. This is considered in the following section.

Etched wiring board design

Defining the etched board wiring is the final major step in the design process. Previous sections have described how the initial design has been successively defined, tested, checked, and packaged into small cards. Since each small card was assigned a position on an etched board, the process has resulted in the gross definition of the interconnections which must be formed between the pins of the etched boards. The primary task of the Etched Board Wiring Design Programs is to compute this wiring in complete detail.

As with the previously described design-assistance programs, the source of data for wiring is the magnetic tape file holding the accumulated logic design. This is of considerable importance since it gives absolute assurance that the wiring and logical design agree. Another important provision of the programs is the ability to retain a history file of the wiring data. Use is made of this file to compute the add/delete wiring data that is necessary to translate from one design level to another.

The succeeding sections of this paper will refer to the computation of the complete wiring for an etched board as "original wiring" design, while the computation of add/delete wiring will be called "rework wiring" design.

• *Physical layout of etched wiring board*

The physical arrangement of the etched board is shown schematically in Fig. 11. The board has two surfaces on which the wiring pattern may be etched. Communication between the two surfaces occurs by means of plated-through holes. Pins occupy some of the plated-through holes and constitute the means by which signals enter or leave the board. Plated-through holes without pins are called "vias" and are used solely to interconnect the surfaces. The holes are arranged in a regular rectangular 65 × 98 matrix, and are grouped in 5 × 14 arrays to form sockets. Each socket contains 3 columns of vias and 2 columns of pins.

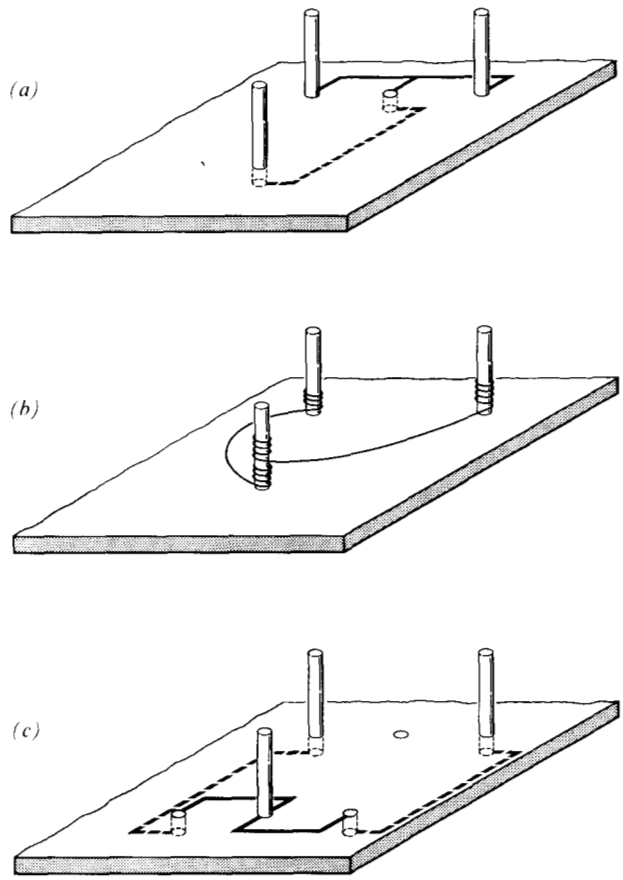


Figure 12 **Methods of connecting a three-pin net.** *(a) multi-ended etched wiring, (b) pin-to-pin discrete wiring, (c) pin-to-pin etched wiring.*

Up to three etched wires are placed in the space between adjacent holes, as illustrated. The etched patterns are produced by a numerically controlled machine which utilizes a moving light source to expose the patterns. Additional connections between pins can be made by discrete wires that are wire-wrapped to pins on the obverse side of the board from the pluggable cards.

• *Original etched wiring computation*

Computation of a set of etched wiring segments which satisfies the interconnection requirements is a demanding task, primarily because of the constraints imposed by the requirement that only two planes be used for the etched wiring.

The following are the significant features of the programs which compute the wiring:

1. Where circumstances require, the engineer may prespecify a wiring route. This is provided so that special cases may be handled. Therefore, the programs first consider any such manually generated data.

2. Etched connections, unlike discrete connections, may be "multi-ended" as illustrated in Fig. 12. Since all vertical wiring segments are kept on one plane and horizontal segments on the other (for efficient space utilization), it is important to make use of these multi-ended connections where possible to decrease wire length and increase wiring density.

3. The pattern of pin connections within a net must sometimes be arranged in severely constrained ways to allow for the characteristics of particular circuits. The programs therefore include a provision for accepting special configuration rules associated with the use of particular small cards.

4. In the computation of the actual wiring paths, two separate methods are employed sequentially. A heuristic approach completes approximately 60% of all required connections. This is accomplished within approximately 2 to 3 minutes of 7090 computer time. A maze-running approach,[6,7] which is exhaustive in finding an open path for a required interconnection, is used to compute the remaining interconnections and completes an average of 95% of all interconnections. This is accomplished in approximately 30 minutes of computer time. The remaining 5% of necessary connections are completed by discrete wire jumpers.

• *Results of wiring trials*

In the course of developing the wiring methods many experiments were made to determine the best approach. Some of the more significant results are as follows:

1. The multi-ended method was better than the pin-to-pin method with respect to average wire length and quantity of discrete wires necessary to complete the wiring. Table 2 compares the results obtained using the two methods.

Table 2 **Comparison of multi-ended vs pin-to-pin wiring results.**

			Discrete wires remaining after wiring	
Board number	*Pins per board*	*Nets per board*	*Multi-ended method*	*Pin-to-pin method*
1	961	314	0	3
2	1161	514	54	68
3	989	323	23	57
4	1113	297	164	203
5	1046	298	52	97
6	901	261	0	13

2. It was found to be a definite advantage to control the order in which various nets were wired. An average of 15% fewer discrete wires are necessary if nets are wired in order of decreasing size rather than in a random order. (The size of a net is defined here as the perimeter of the smallest rectangle which contains all the pins of a net.)

3. Computer processing time and the required number of discrete wires both increased significantly when the maze-running technique was used alone rather than in conjunction with the heuristic technique.

4. Restrictions placed on the freedom of the maze-running program result in a measurable decrease in the number of discrete wires required. These restrictions consist of limiting the area in which the program is allowed to operate when searching for open paths, and of limiting the number of vias and the length of printed conductors permitted to complete a path. Best results are obtained by initially limiting the program while attempting all the required connections, and gradually relaxing the restrictions in subsequent iterations.

The Etched Wiring Board Design Programs used for System/360 designs are arranged to employ the above results to best advantage. As one might expect, invested computer time follows a curve of diminishing returns. The parameters which control the restrictions and iterations can be adjusted by control-card input for each board processed. Table 3 shows the results when the wiring paths for boards of varying degrees of complexity are computed using the three available sets of control parameters.

• *Rework wiring computation*

Following the completion and release of an initial design, the engineer is faced with the problem of incorporating design changes. It is often necessary that these changes be treated as modifications to the existing design, rather than as completely new designs.

For this purpose the Etched Wiring Board Design Programs have the ability to derive the logical additions and deletions to be made to a previous design level. The logical changes are then applied to the previous wiring configuration, modifying it into the desired new design.

The significant characteristics of this process are:

1. Only the unwanted wiring in the original design is deleted.

2. The additional wiring required is added such that a complete new design is available for the manufacturing process. For efficiency, this process demands maximum use of etched connections.

3. Both of the above processes are completed under the restraint that the deletion/addition steps can be described

Table 3 **Results of wiring methods.**

Board characteristics		*Results*	*1*	*N*	*2*
Board 1		7090 running time (in minutes)	12	4	0.5
Number of pins	791	Number of discrete wires	12	16	60
Number of nets	280	Number of vias used	915	898	718
Average pins per net	2.32	Average vias per net	3.36	3.34	3.01
		Average inches per net	6.75	6.77	6.85
Board 2		7090 running time (in minutes)	37	18	2
Number of pins	1050	Number of discrete wires	60	78	139
Number of nets	357	Number of vias used	1319	1179	852
Average pins per net	2.94	Average vias per net	4.27	3.94	3.37
		Average inches per net	8.64	8.69	9.11
Board 3		7090 running time (in minutes)	94	60	7
Number of pins	1299	Number of discrete wires	169	188	272
Number of nets	362	Number of vias used	1653	1576	1099
Average pins per net	3.58	Average vias per net	5.66	5.69	4.70
		Average inches per net	10.72	11.15	11.62

Legend
1 Parameter set for use when a larger-than-normal investment in computer time is warranted.
N Set recommended for normal use.
2 Parameter set resulting in minimum computer time.

and performed on a previously manufactured board by an efficient manual process. Manual deletions are accomplished by severing existing etched wiring or removing existing discrete connections; additions are accomplished by adding discrete wire.

4. As a result of the two physical routes by which a given board may be obtained (i.e., newly manufactured or manually reworked), the entire process is constrained so that any future changes can be implemented identically on the two physical boards. This is accomplished by making all additional etched connections in a pin-to-pin manner and with direct correspondence to the discrete connections added manually.

In addition to being necessary for the efficient manufacturing and field servicing of machines which are subject to design changes, the rework programs offer two important design benefits: they enable design iterations to be made under tightly controlled conditions, and they significantly reduce the computer time required for producing a modified design. (A rework design change which affects 10% of the wiring can be made in 30% of the computer time required to recompute the entire design as an original.)

Conclusions

The use of Solid Logic Design Automation has materially assisted in the application of Solid Logic Technology to the new System/360 computer family. Specifically, Solid Logic Design Automation:

1. Has provided the various laboratories participating in the project with identical design procedures for the System/360.

2. Has provided uniform documentation so that all plants can manufacture designs from any laboratory and anyone trained on one machine can understand the documentation of any other.

3. Has provided designers with powerful new tools for organizing, validating, and detailing their designs with a previously unattainable degree of accuracy. This increased accuracy is particularly valuable because packaging on etched boards imposes severe penalties for mistakes found *after* prototypes are constructed.

4. Has made it easy for designers to evaluate design variations in order to select the best alternative.

5. Has made possible the economical design of etched wiring boards and flat cables.

6. Has converted logic diagrams into digital information to control automated tools that manufacture etched wiring boards and flat cables.

Acknowledgments

It is impractical to mention all the individuals whose programming contributions made possible the implementation of this unified set of programs—a programming task ten times the magnitude of FORTRAN. However, important concepts were contributed by R. R. Burch, D. J. Galletta, V. A. Nelson, R. J. Preiss, J. E. Steitz, S. G. Tucker, and R. H. Glaser. The Printed Circuit Generator referred to in this paper was developed at the IBM Manufacturing Research Laboratory in Endicott, N. Y. The support and encouragement of E. Bloch and D. L. Kilcrease proved invaluable. Special acknowledgment is due to D. R. White for his assistance in the preparation of this paper.

References

1. M. Kloomok, P. W. Case, and H. H. Graff, "The Recording, Checking, and Printing of Logic Diagrams," *Proceedings of the EJCC*, 108 (December, 1958).
2. G. A. Blaauw, G. M. Amdahl, and F. P. Brooks, Jr., "Architecture of the IBM System/360," *IBM Journal* 8, No. 2, 87-101 (April, 1964).
3. E. M. Davis, W. E. Harding, R. S. Schwartz, and J. J. Corning, "Solid Logic Technology: Versatile, High-Performance Microelectronics," *IBM Journal* 8, No. 2, 102-114 (April, 1964).
4. R. C. Paulsen and W. K. Springfield, "High-Frequency, Multiple-Signal-Conductor Transmission Line." In preparation.
5. C. H. Haspel, "Automatic Packaging of Computer Circuitry." In preparation.
6. U. R. Kodres and H. E. Lippman, "SLT Board Layout," IBM Technical Report TR00.1010, Revised March 10, 1964.
7. C. Y. Lee, "An Algorithm for Path Connections and Its Applications," *IRE Trans. on Elect. Computers* EC-10, No. 3, 346-365 (September, 1961).

Received January 20, 1964