A COMPUTER AIDED OPTIMIZATION TECHNIQUE FOR DIGITAL SYSTEMS TIMING

Report 1-66-42

By

J. E. Palmer

Dr. H. W. Mergler
Professor of Engineering
Principal Investigator
NASA GRANT NsG 36-60

1966

ERRATA

P 47 Figure 4.1 S1 and S2 should be labeled RST

P 49 A space is required between SXX and (ϕ)

P 50 A space is required between PXX and (ϕ)

P 54 A double space is required after the symbol G in the G Card Format. Also in Figure 4.5 (a) the amplitude of P (σ) should be 1/20 rather than 1/40.

P 80 Figure 4.18 the NO exit on the decision block for "N"? goes to ERRO.

P 120 The YES exit on $(MATL)_{16}$=0? goes to AND, the NO exit goes to OR.

P 126 The YES exit from both TBIT=0? decision blocks goes to ANDX. The two boxes with no exits go to EXIT.

P 133 Figure 4.30. A line should be added from TINP to the decision block immediately below. The NO exit from the decision block ACC=1? goes to EXIT.

P 183 Figure 6.3. The labels on the cross hatching are reversed.

P 210 The first line should read Table 7.5.

P 225 Figure A-1 (a). $P_x(t)$ should be a square pulse between 300 and 400.

ABSTRACT

A brief summary of worst case and statistical design techniques is given along with a discussion of the applicability of these techniques to the design of digital systems using microcircuits. A technique for statistical optimization using component sorting on the basis of propagation delays is presented and a few examples of the expected improvement are shown. A discussion of the available technique for evaluation of statistical design results in the evaluation of a digital computer program for the DDP-116 computer.

This computer program, using Monte Carlo techniques and a simulation of the system under test, is designed to be a useful engineering tool. It requires no specialized knowledge on the part of the designer, other than his ability to read logic diagrams and component specification sheets.

Some design examples, using the statistical techniques and the computer evaluation, are used to demonstrate the applicability to digital system design.

Table of Contents

List of Figures

List of Tables

CHAPTER I

INTRODUCTION

1.1 The Design Problem

The art of design is far less precise, generally, than the science of analysis. The major difference is in the nature of the problem. In analysis the problem is usually closed, that is, the system or circuit is already constructed, at least conceptually, and its behaviour is fixed by constraints. On the other hand, the nature of the design problem is open, the desired performance only is often specified, and the solution to the problem is only limited by the experience and ingenuity of the designer.

It is, however, often possible to divide a design problem into two phases. The first phase, where the system or circuit configuration is determined by the designer, is mainly a creative phase and, as such, does not lend itself to formalization. The second phase is generally one of optimization with regard to some set of parameters. Most of the so-called design techniques are really optimization techniques when viewed in this context. It is a proposed optimization technique which is the subject of this thesis.

1.2 Worst Case Design Techniques

An excellent and widely used design technique is "worst case design." Worst case design has been defined by Pressman[1] as "designing so that the circuit will still work when all supply voltages and active and passive components are simultaneously at those extremes of their tolerance limits that will tend most to make the circuit inoperative". Pressman's definition was meant for circuits, but with appropriate changes in parameters could equally apply to systems. Most digital systems, designed today, in fact, use worst case design on all timing considerations.

In practice, the worst case design criteria must be slightly modified upon consideration of correlations between parameter variations. For example, the use of a common supply voltage bus precludes using at one extreme in one portion of a system and at the other extreme in a different portion. Again, consideration of the effect of thermal variations must allow for correlation in parameter changes. A fine illustration of this effect is found in "Transistor Circuit Design".[2]

Worst case design techniques are widely used and relatively easy to apply; however, they are subject to limitations which may be important in many design environments.

First, in many cases of interest, there is no solution to the worst case design equations unless extremely tight tolerance limits are imposed on the active and passive components. The need for precise components causes increased component cost. This limitation raises an interesting question, which will be discussed at length in this thesis, "Are there more effective ways to sort components than on the basis of tolerance limits?".

Secondly, worst case design procedure only assures no failures in the completed equipment if the component suppliers have met their tolerance limits and if there have been no so-called catastrophic failures in the production process. Put differently, worst case design does not guarantee immunity from trouble shooting and de-bugging.

Third, there is no knowledge obtained from a worst case design about the amount of penalty to be paid for leaving the "safe design area". All that is known is that the possibility of failure exists; nothing is determined about the probability of failure.

Finally, there is no assurance that a worst case design is optimum in all respects. In particular, Ware and Becker[3]

discuss the point that a worst case design may actually yield a circuit which is less than optimum from the standpoint of reliability (mean time between failures).

1.3 Statistical Design Techniques.

Another approach to design optimization is statistical analysis of the system in question. As presently utilized, statistical design is not so much an alternative to worst case design as it is a supplement to it. This point is aptly illustrated in Professor Gray's "Digital Computer Design"[4] which contains, as well, a very lucid discussion of statistical design techniques in general.

Briefly, the statistical distributions of the various component parameters are assumed known; then in many cases standard analytical techniques are used to transform the parameter variations into system output statistics. In other cases, it is necessary to model the system and apply Monte Carlo techniques to obtain the same information.

The major advantage claimed for statistical design is that it allows the establishment of trade-offs between component tolerance, system performance, and failure rates in fabrication. These trade-offs when interpreted in a particular design

environment (economic factors, reliability, etc.) allow the designer much more flexibility in fulfilling his goals.

The major objections to statistical design fall into two categories.

(a) The data is unavailable. The statistical data required is not available from the component supplier or it is in such a form as to be unyielding to statistical techniques. The data supplied by the manufacturer is also rapidly obsolete as the production process is modified. This unavailability of data is particularly pronounced with active components. It should not, however, be inferred that this data is impossible to obtain; it is expensive to obtain in many cases and there has been no real demand for it on a large scale.

(b) Considerable expense is involved. In all but the most trivial cases, statistical analysis requires the use of a digital computer and the associated costs are significant. Since the cost of the analysis is spread over the entire output, its use in low volume production cannot often be justified economically. It is worth noting in this connection that the lease or purchase of a digital computer is not always necessary. At least one

firm[5] offers to subcontract the statistical analysis of circuits in its entirety.

It is true, however, that any consideration of the applicability of statistical design must be viewed in the light of these two objections.

1.4 Microcircuitry Considerations

One of the most dramatic advances in the field of digital systems has been the development of monolithic silicon microcircuits. From a relatively exotic beginning in the missile and space industry, the microcircuit has become a widely used digital system component in only five years.

Detailed discussions of microcircuits are readily available in the literature; several journals[6, 7] have devoted entire issues to them. For the purposes of this report, their main properties are summarized below:

(a) Small size and low weight.

(b) High speed.

(c) Reliability. (A good bit of this is extrapolated transistor data.)

(d) Relatively high tooling costs for new circuit designs.

(e) Difficulties in obtaining log delays. The monolithic circuits obtain the delays from p-n junction capacitance. For this reason it is difficult to obtain values in excess of picofarads.

(f) A rather large spread in the values of propagation delays. From representative manufacturers' specification sheets the spread in propagation is usually an appreciable portion of the mean propagation delay, usually at least ± 25%, often times considerably more. This spread can be reduced by selection but at added expense to the user.

Properties a, b, and c listed above indicate that microcircuits are particularly well suited for use in digital systems. The high tooling costs mean that utilization is restricted to high volume production for new circuit designs or to the use of commercially available circuits. In either of these cases, the high tooling cost is spread over many thousands of units. The difficulty in obtaining long time delays can usually be overcome by a hybrid approach using external capacitors. It is reasonable then to expect that an increasing proportion of digital systems will be composed of microcircuits.

Industry clearly shares this view. The IBM 360 computer, already on the market, uses the hybrid microcircuits (thin films for passive components, silicon chips for active elements).

Discussions with representatives of other computer manufactures indicate a consensus that the next generation of computers will be almost entirely composed of microcircuits.

A projected usage curve for microcircuits was shown at the 1964 IEEE International Convention. This report[8] represents industry thinking near the end of 1963 and is shown in Figure 1.1 Viewed from the vantage point of 1966, this estimation was not overly optimistic. In particular, the prophesy[8] of Maier that "In high volume areas, where standardization is feasible, semiconductor circuits may equal the cost of conventional discrete assemblies in the next few years", has been essentially fulfilled.

The large spread in propagation delays mentioned earlier becomes increasingly important when the system design using microcircuits is considered. This design differs from design with conventional components in that the number of parameters are fewer. Instead of resistors, transistors, diodes, etc., the circuit is the component and the parameters of interest to the designer are:

a) fan in and fan out limits

b) rise and fall times

c) power dissipation

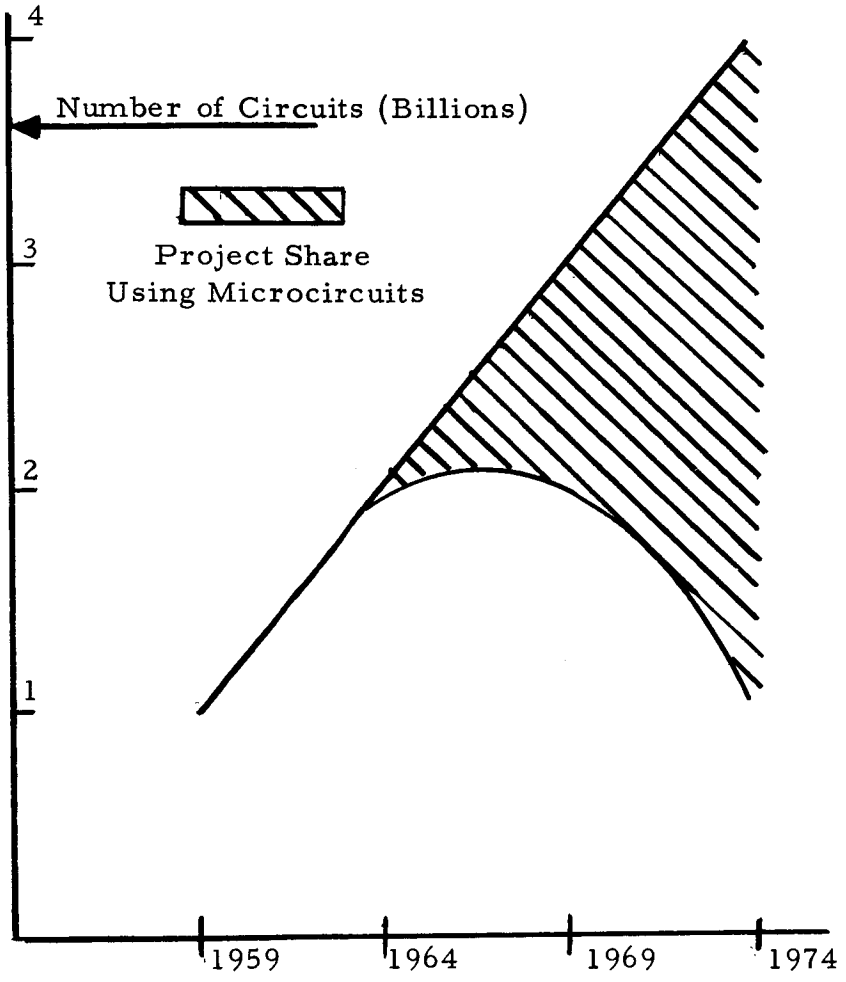


Figure 1.1 Projected Use of Microcircuits. (From "Integrated Circuits in Industrial Equipment" Leonard C. Maier. IEEE Spectrum Vol. 1, No. 6, June, 1964.)

d) propagation time

In analogy to circuit design, where the object was the optimum choice of discrete components for a specific circuit job, the system design with microcircuits has for a goal the optimum selection of circuits to form a system block. This is a problem to which the design techniques previously discussed can be applied. In particular, the area of interest to this thesis is the application of statistical design techniques to the transient response of digital systems. The wide spread in propagation delays of an individual microcircuit indicates that such an approach might yield significant improvement in system performance or significant reduction in system cost for a given performance.

1.5 Component Sorting

A widely used technique in the fabrication of electronic equipment is the selection of components. In this case, a component which meets almost all the requirements set for it is selected, then, as the components are manufactured, it is tested and only those units meeting all of the specifications are used in the equipment. An example would be the choice of a transistor of a given type, but only the acceptance of those units having an alpha

cutoff frequency in excess of 10 mc. This results in a higher unit price for the components because of the added testing required and because of the lowered effective yield. This higher price can often be economically justified by a decrease in the complexity of the associated circuitry.

A related technique which has not had such widespread use is that of sorting components. In this case the components are all accepted but are divided into different classes on the basis of some parameter. Note the distinction, selection implies that some components originally produced are not used, sorting implies all are used.

An obvious application of sorting is to make those components in critical places the members of the best class of the sort, putting the others into non-critical portions of the system. However, it is pointed out in Chapter II that the mere act of sorting, with no attempt to place components in any special order, can improve the performance of a system.

It is then the purpose of this paper to investigate the applicability of statistical design techniques to the transient response of digital system, particularly the applicability of

component sorting on a statistical basis, and to develop a useful engineering tool for the application of these techniques.

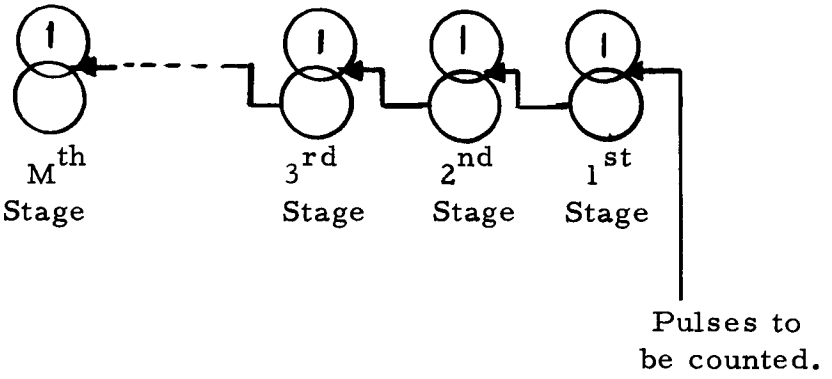
CHAPTER II

THE EFFECT OF COMPONENT SORTING ON THE TIMING PERFORMANCE OF A DIGITAL SYSTEM

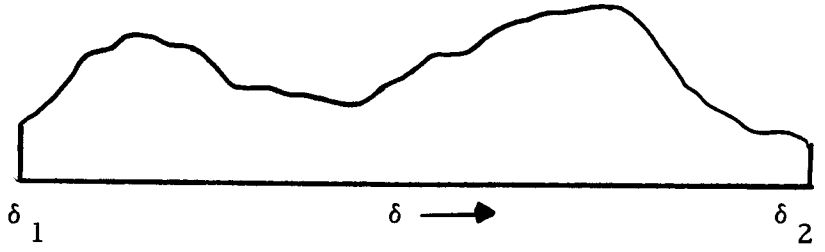
2.1 Identical Elements in Sequence

In many important applications, identical elements are triggered sequentially. An excellent illustration is a binary counter, as shown in Figure 2.1(a). Here the maximum rate at which the counter can be used is the ripple time, that is, the time it takes for the counter to change from all 1's to all 0's. This time is the sum of the propagation times of each of the individual counter components. Suppose that the elements are all of the same type and that the propagation delay of each component is specified by a probability density function as shown in Figure 2.1(b). The maximum time delay is δ_2 and the minimum delay is δ_1. If the counter contains M stages then the maximum possible ripple time is clearly $M\delta_2$ and the range of possible ripple times is $M(\delta_2-\delta_1)$.

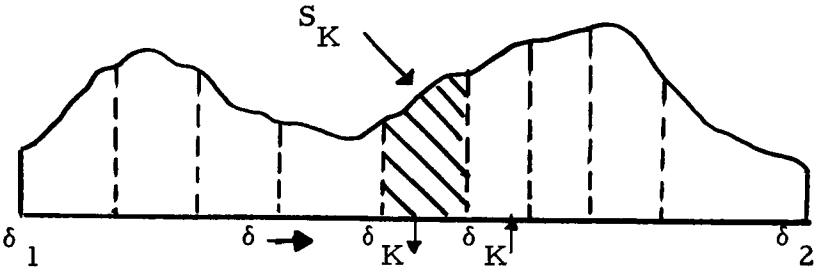
Suppose now that the total population of the element type are sorted into M sets, δ_k, such that each of the sets has the same population. An element is put into set δ_k if its propagation delay is less than that of any element in set δ_{k+1} and greater than that of δ_{k-1}. To do this conveniently would require a knowledge of the propagation delay density function.



(a) A Binary Counter (M Stages)



(b) A Possible Density Function of Time Delays for a Counter Element.



(c) A Possible Sort of the Function in (b)

Figure 2.1 Hypothetical Delay Parameters for a Binary Counter.

The ripple counter is then constructed by choosing at random one element from each of the δ_k. Now the longest possible ripple time is given by the sum of the longest delays in each group. Because the longest delay in the set, δ_M, with the longest delays is δ_2, and because all the groups contain delays less than this number, the new longest possible ripple time is less than $M\delta_2$, the value obtained in the unsorted case. The amount of decrease will depend on the shape of the probability density function, but the fact that a decrease occurs depends only on the fact that $\delta_2 > \delta_1$.

$$\text{Range} = \delta'_{max} - \delta'_{min} = \sum_{k=1}^{M} \delta_k\uparrow - \sum_{k=1}^{M} \delta_k\downarrow \qquad (2.1)$$

where δ'_{max} = longest possible ripple time

δ'_{min} = shortest possible ripple time

$\delta_k\uparrow$ = longest delay in set k

$\delta_k\downarrow$ = shortest delay in set k

If the original density function were at least piecewise continuous then:

$$\delta_k\uparrow = \delta_{k+1}\downarrow \qquad (2.2)$$

and

$$\text{Range} = \delta_{M}\uparrow - \delta_{1}\downarrow = \delta_{2} - \delta_{1} \tag{2.3}$$

The effect of sorting has been to reduce the range of possible ripple times by a factor of 1/M.

The mean time delay of an unsorted element will be defined as μ. If the delays of the elements in the counter may be assumed statistically independent of each other then, the mean ripple time is $M\mu$. This is also true of the case with sorting as the following analyses show.

If the delays of the elements are statistically independent the mean of the sum of delays (ripple time) is equal to the sum of the means or;

$$\mu' = \sum_{k=1}^{M} \int_{\delta_k\downarrow}^{\delta_k\uparrow} \delta\, p_k(\delta)\, d\delta = \sum_{k=1}^{M} \mu'_k \tag{2.4}$$

where $p_k(\delta)$ is the probability density function of delays in the $k^{\underline{th}}$ set.

Note that

$$p_k(\delta) = M\, p(\delta) \qquad \delta_k\downarrow < \delta < \delta_k\uparrow$$
$$= 0 \qquad \text{elsewhere} \tag{2.5}$$

If we note the substitution

$$\mu' = \sum_{k=1}^{M} M \cdot \int_{\delta_k}^{\delta_k} \delta\, p(\delta)\, d\delta = M \int_{\delta_1}^{\delta_2} \delta\, p(\delta)\, d\delta \tag{2.6}$$

$$= M\mu$$

Thus, the mean ripple time is not affected by the act of sorting, but the range of ripple times is reduced.

A qualitative insight into the behaviour of the variance of the ripple time can be obtained by noting that for statistically independent delays among the elements, the unsorted variance is

$$\sigma^2 = \sum_{k=1}^{M} \int_{\delta_1}^{\delta_2} (\delta - \mu)^2\, p(\delta)\, d\delta = M \int_{\delta_1}^{\delta_2} \delta^2 p(\delta)\, d\delta - M\mu^2 d\delta \tag{2.7}$$

while for the case of the sorted components

$$\sigma'^2 = \sum_{k=1}^{M} \int_{\delta_k\downarrow}^{\delta_k\uparrow} (\delta - \mu'_k)^2 \, p_k(\delta) \, d\delta \qquad (2.8)$$

$$\sigma'^2 = \sum_{k=1}^{M} \int_{\delta_k\downarrow}^{\delta_k\uparrow} \delta^2 \, p_k(\delta) d\delta - \sum_{k=1}^{M} (\mu'_k)^2$$

Where in (2.8) use is made of the fact that from the definition of the mean

$$\mu'_k \int_{\delta_k\downarrow}^{\delta_k\uparrow} \delta \, p_k(\delta) \, d\delta = (\mu'_k)^2 \qquad (2.9)$$

Using relation (2.5), it follows that

$$\sigma'^2 = M \int_{\delta_1}^{\delta_2} \delta^2 \, p(\delta) \, d\delta - \sum_{k=1}^{M} (\mu'_k)^2 \qquad (2.10)$$

In order to evaluate the relative amounts of the variances in the two cases the difference of equations (2.7) and (2.10) is

formed:

$$\sigma^2 - \sigma'^2 = -M\mu^2 + \sum_{k=1}^{M} \mu'^2_k \qquad (2.11)$$

To evaluate the sign of equation (2.11), use is made of an inequality due to Cauchey which states:

$$\left(\sum_{i,j=1} a_i b_i \right)^2 \leq \sum_{i=1} a_i^2 \sum_{j=1} b_j^2 \qquad (2.12)$$

letting $a_i = \mu_i \qquad b_i = 1/M$

$$\left(\sum_{i=1}^{M} \mu_i \frac{1}{M} \right)^2 \leq \left(\sum_{i=1}^{M} \mu_i^2 \right) \left(\sum_{j=1}^{M} \frac{1}{M^2} \right) \qquad (2.13)$$

in this case $\sum_{i=1}^{M} \mu_i = \mu M$, then

$$\mu^2 \leq \frac{1}{M} \sum_{1}^{M} \mu_k^2 \qquad (2.14)$$

Thus the effect of sorting is always to lower the variance, effectively "tightening" the ripple time density function about the mean. It is important to note that "identical" in the sense used here refers only to the fact that the same statistics

govern the propagation delay.

2.2 A Numerical Example

To gain a quantitative insight into the properties of sorting, a few numerical examples were attempted. For simplicity in computation the uniform probability density function was assumed for $p(\delta)$.

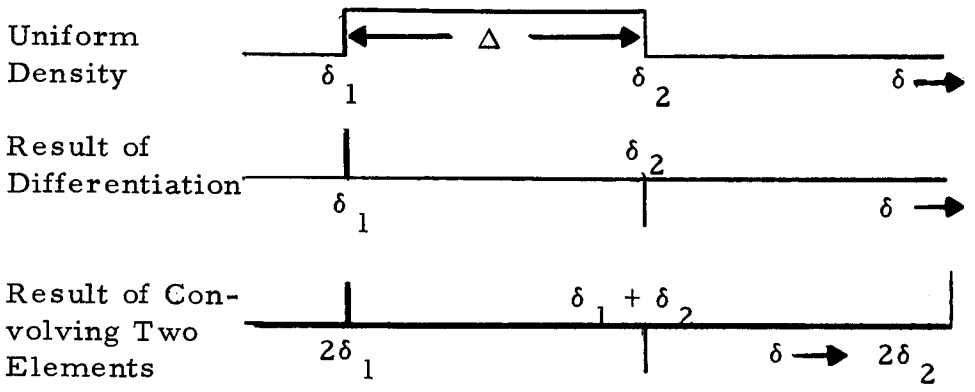
Referring to Figure 2.2(a) and using the method of impulse convolution described in Appendix A, the $M^{\underline{th}}$ derivative of the ripple time density function for the unsorted case was found to be:

$$f^{M} = \frac{1}{\Delta^{M}} \sum_{j=1}^{M} \frac{(-1)^{j} M!\ I(\delta - Ma - j\Delta)}{(M-j)!\ j!} \qquad (2.15)$$

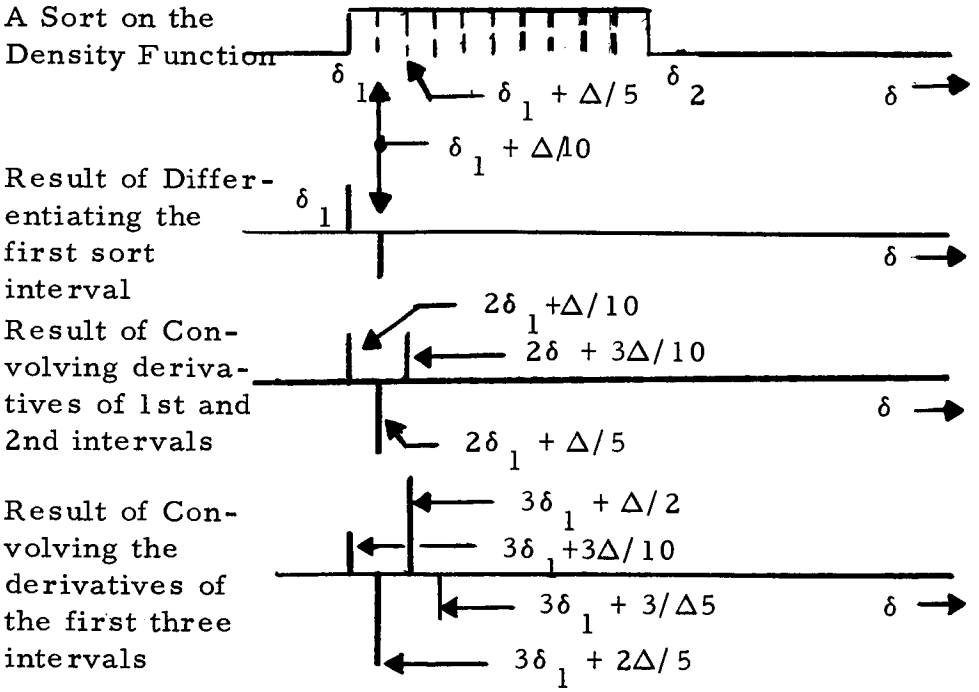
where in order to avoid confusion the symbol $I(x)$ is used to denote the unit impulse occurring at x rather than the customary $\delta(x)$.

Integration of (2.15) yields the density function of the ripple time

$$f = \frac{1}{\Delta^{M}} \sum_{j=1}^{M} \frac{(-1) M (\delta - Ma - j\Delta)^{M-1} U(\delta - Ma - j\Delta)}{(M-j)!\ j!} \qquad (2.16)$$



(a) The Unsorted Elements



(b) The Sorted Elements

Figure 2.2 Illustration of Impulse Convolution

Here U(x) is the unit step occurring at x.

Another integration yields for the probability distribution function.

$$f^{-1} = \frac{1}{\Delta^M} \sum_{j=0}^{M} \frac{(-1)^j(\delta - Ma - j\Delta)^M \, U(\delta - Ma - j\Delta)}{(M-j)!\; j!} \qquad (2.17)$$

Referring now to Figure 2.2(b), and using the same method, the density function for the sorted case is

$$f'^{M} = \frac{1}{\Delta'^{M}} \sum_{j=0}^{M} \frac{(-1)^j M!\; I(\delta - Ma - \Delta' [\sum_{k=0}^{M-1} k+j])}{(M-j)!\; j!} \qquad (2.18)$$

Recalling that $\Delta' = \Delta/M$, and integrating to find the ripple time density function

$$f' = (\frac{M}{\Delta})^M \sum_{j=0}^{M} \frac{(-1)^j M!\; (\delta - Ma - \frac{\Delta}{M}[\sum_{k=0}^{M-1} k+j])^{M-1} U(\delta - Ma - \frac{\Delta}{M}[\sum_{k=0}^{M-1} k+j])}{(M-j)!\; j!\; (M-1)!} \qquad (2.19)$$

And the density function is then given by

$$f'^{-1} = (\frac{M}{\Delta})^M \sum_{j=0}^{M} \frac{(-1)^j(\delta - Ma - \frac{\Delta}{M}[\sum_{k=0}^{M-1} + j])^M \, U(\delta - Ma - \frac{\Delta}{M}[\sum_{k=0}^{M-1} + j])}{(M-j)!\; j!} \qquad (2.20)$$

Figure 2.3 shows a comparison of the density function between sorted and unsorted cases for various values of M. In the case of the uniform density, the effect is to compress the density function about the mean by a factor of M. This is true because the densities of the sorted elements are of the same shape as the original density function. This is not a general case, however.

The results of sorting components with a uniform propagation delay density is shown in Figure 2.4. To obtain these graphs, the probability that the ripple time delay would exceed the ordinate value was obtained for both the sorted and unsorted systems. The difference between the two is a measure of the improvement in performance obtained by the sorting operation. This difference could be considered as that proportion of circuits which were now made useable after sorting, if the maximum time delay were fixed at that ordinate value.

2.3 Sorting by Twos

As a practical matter, it should be noted that the difficulties in making a sort into M sets, increase as M is increased and as the complexity of the shape of $p(\delta)$ increases. As mentioned in the introduction, the objections to statistical analysis

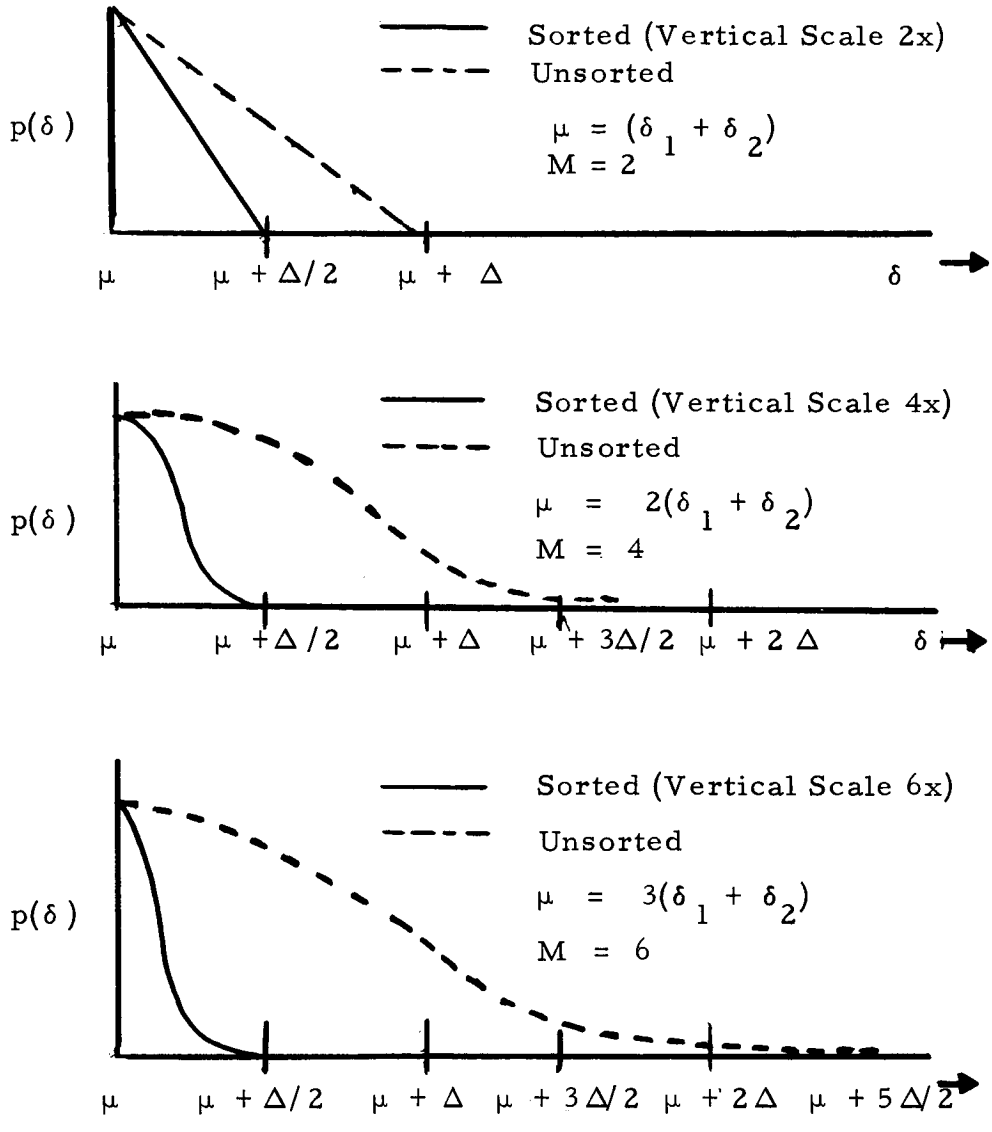


Figure 2.3 Ripple Delay Density Functions for Various Number of Counter State (M), Uniform Element Delay Density Assumed.

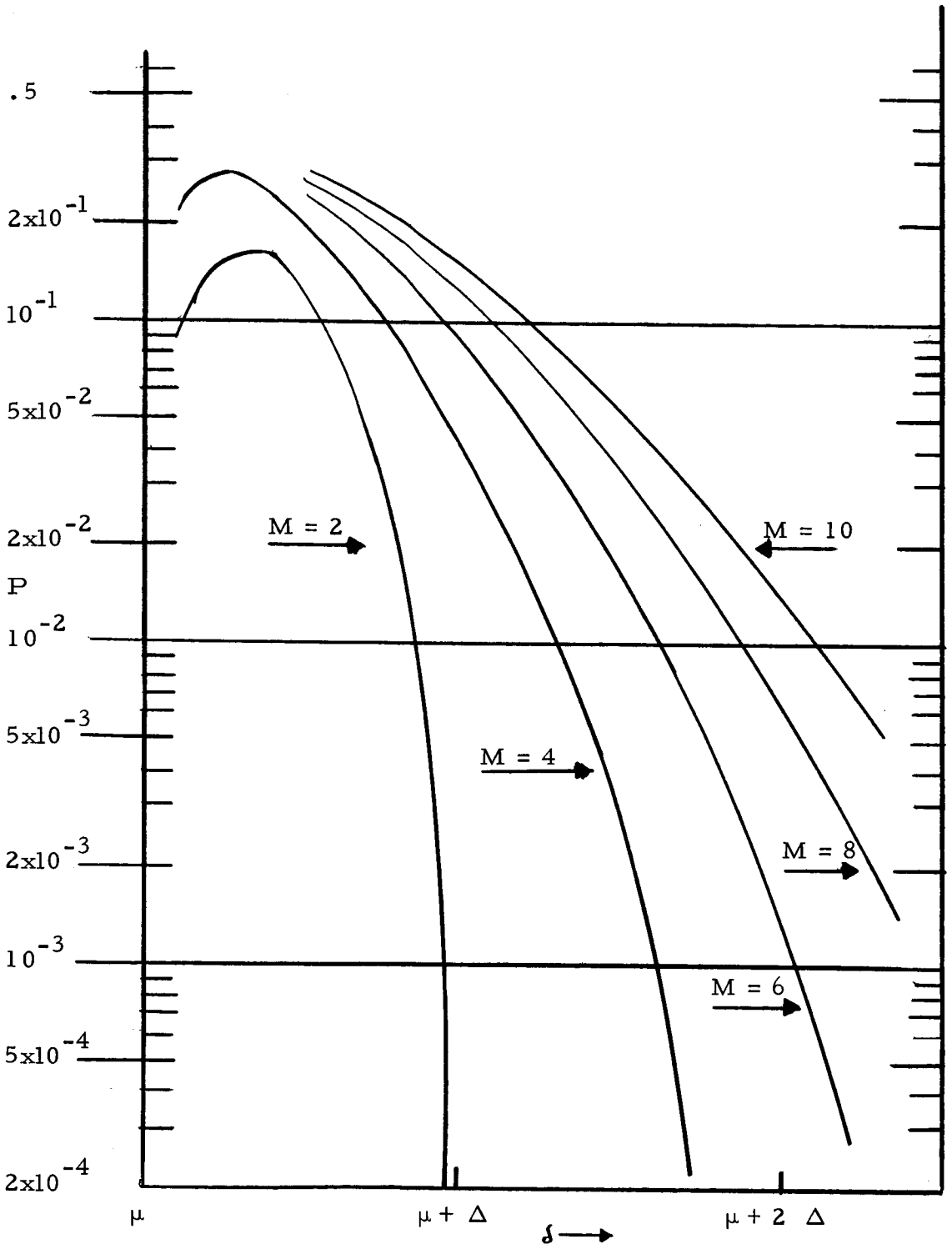


Figure 2.4 Performance Improvement Due to a Complete Sort for Various Numbers of Counter Element (M), Uniform Density Assumed.

on the basis of unavailability and changeability of statistical data are not without validity. It would be desireable to keep the sort as simple as possible.

For this reason, the improvement obtainable by making a sort into only two sets was investigated. The mean value of the component distribution is the only statistic needed here; this is comparatively easy to obtain.

As was seen earlier in this section (although not stated in exactly this form), the spread in ripple time is equal to the sum of the widths of density functions being convolved. Clearly a sort by two's results in a decrease of ripple times by a factor of one half, regardless of the number of elements in the rippling chain.

If there are an even number of elements in the rippling chain of circuits, the mean will be unchanged as long as the number of circuits from each of the two sets is the same. This is easily proven by noting the following steps:

(a) Consider the circuits pairwise with one from each set. Equation (2.6) states that the sum of these two means is twice the mean of the unsorted density function.

(b) Add the means of the M/2 pairs considered in (a). The mean of the sum is equal to the sum of the means. Thus the mean of M elements sorted by two's is the same as the mean of M unsorted elements.

If M is odd and there is one excess element, the mean delay of the M cascaded elements will be less than, equal to, or greater than the mean delay of M unsorted elements in so far as the excess element is chosen from the lower sort, unsorted elements, or upper sort, respectively.

A qualitative insight into the behaviour of the variance under a sort by two's can be obtained in much the same manner as the behaviour of the mean was obtained. First, if an even number of cascaded elements is involved, equation (2.11) can be applied to each pair yielding

$$\sigma'^2 = \sigma^2 - \mu_+^2 - \mu_-^2 + 2\mu^2 \qquad (2.21)$$

In this case σ^2 is the variance of two unsorted elements in cascade $\sigma^2 = 2\sigma_D^2$.

σ_D^2 is the variance of the unsorted elements.

μ_+ is the mean of the upper sort.

μ_- is the mean of the lower sort.

μ is the mean of the unsorted elements.

If the M/2 pairs are now cascaded, the variance of the sum is equal to the sum of the variances. Here is the result:

$$\sigma_2'^2 = M\sigma_D^2 - \frac{M}{2}(\mu_+^2 + \mu_-^2) + M\mu^2 \qquad (2.22)$$

Now $\sigma_2'^2$ is the variance of the chain of M elements.

If a chain of M unsorted elements is considered, the variance will be $M\sigma_D^2$. Equation (2.22) can be rewritten as

$$\sigma_2'^2 = \sigma^2 - \frac{M}{2}(\mu_+^2 + \mu_1^2) + M\mu^2 \qquad (2.23)$$

where for purposes of comparison σ^2 now means the variance of the M unsorted cascaded elements.

If equations (2.11) and (2.23) are compared they differ only by the quantity

$$\sigma_2'^2 - \sigma'^2 = \sum_{k=1}^{M} \mu_k^2 - \frac{M}{2}(\mu_+^2 - \mu_-^2) \qquad (2.24)$$

In Appendix B it is shown that equation (2.24) represents a positive quantity. Thus the act of sorting by two's reduces the variance of the ripple time, but by a lesser amount than does the complete sort.

With the foregoing in mind, the numerical example of the uniform distribution was repeated, using this time a sort by two's. The results are plotted in Figure 2.5. No curve is shown for M = 2, since in this case a sort by two's is a complete sort and the curve is identical to the M = 2 curve in Figure 2.4. Note that in regions where the probability of a delay is zero for both the sort by two's and the complete sort, then the performance improvement is equal for the two cases.

The data can be viewed in another light. Instead of considering the improvement in performance by comparing probabilities at a given time delay, a useful comparison is obtained by comparing time delays at a given probability. In this way a measure of the time saved can be obtained. Both the complete sort and the sort by two's were compared with the unsorted delay at various probabilities. The difference was called the time saved. Figure 2.6 shows the time saved by the two methods. As a rule of thumb, for the uniform density function assumed here, the sort by

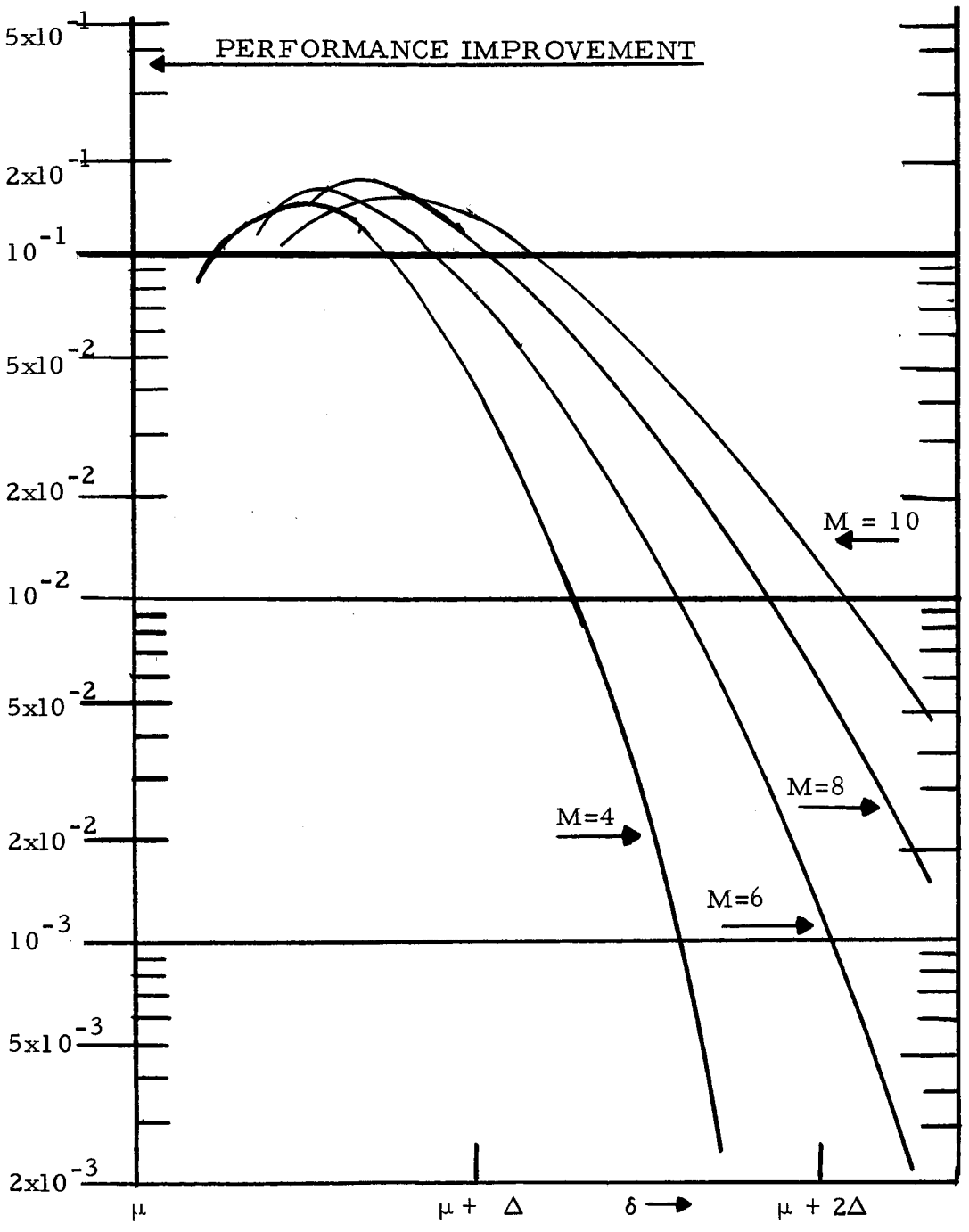


Figure 2.5 Performance Improvement Due to a Sort by Two's for Various Numbers Of Counter Element (M) (Uniform Density Assumed).

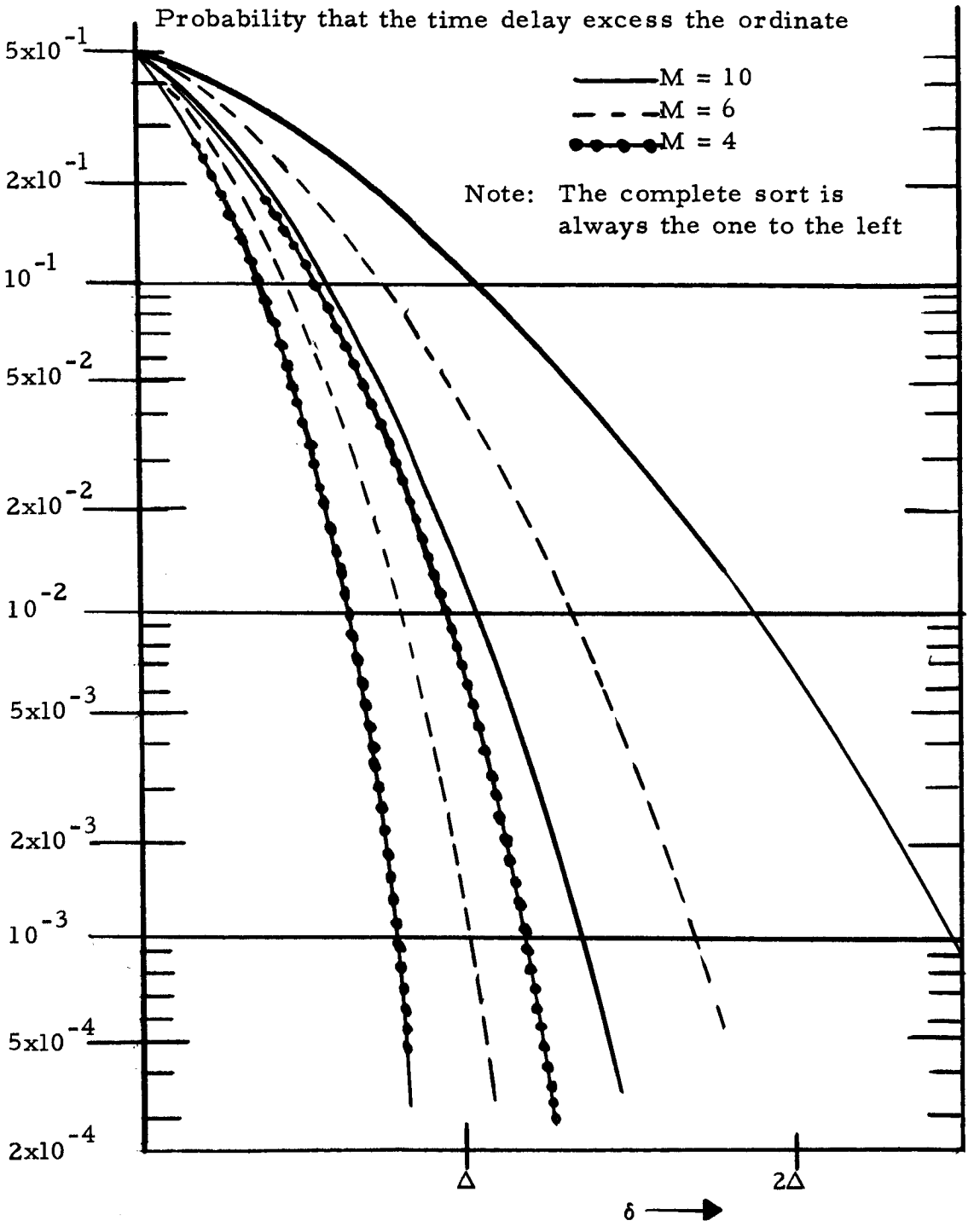


Figure 2.6 Comparison of Time Saved Between Complete and Sort by Two's for Various Values of M.

two's achieves a saving of from 55% (M=10) to 75% (M=4), obtainable with a complete sort.

In an effort to determine whether the sort by two's achieves the same results for other densities, the time saved by a complete sort was compared with the time saved by a sort by two's for a triangular density function for chains of length 2, 4, 6, 8, and 10. In this case the sort by two achieved about 65% of the saving obtainable with a complete sort.

It should be noted that this later data was obtained by Monte Carlo simulation while that for the uniform density was obtained by the mathematical analysis. Figure 2.7 shows a representative data curve (M=4) from which the estimates of time saving were made. To ensure the accuracy of the Monte Carlo approach, several check points were obtained by analytical techniques. No significant departures from the Monte Carlo values were observed.

2.4 Non-Identical Elements in Sequences

In many cases of practical importance a pulse is propagated through a chain of non-identical elements. That is, the propagation delay of the elements in the chain is governed by

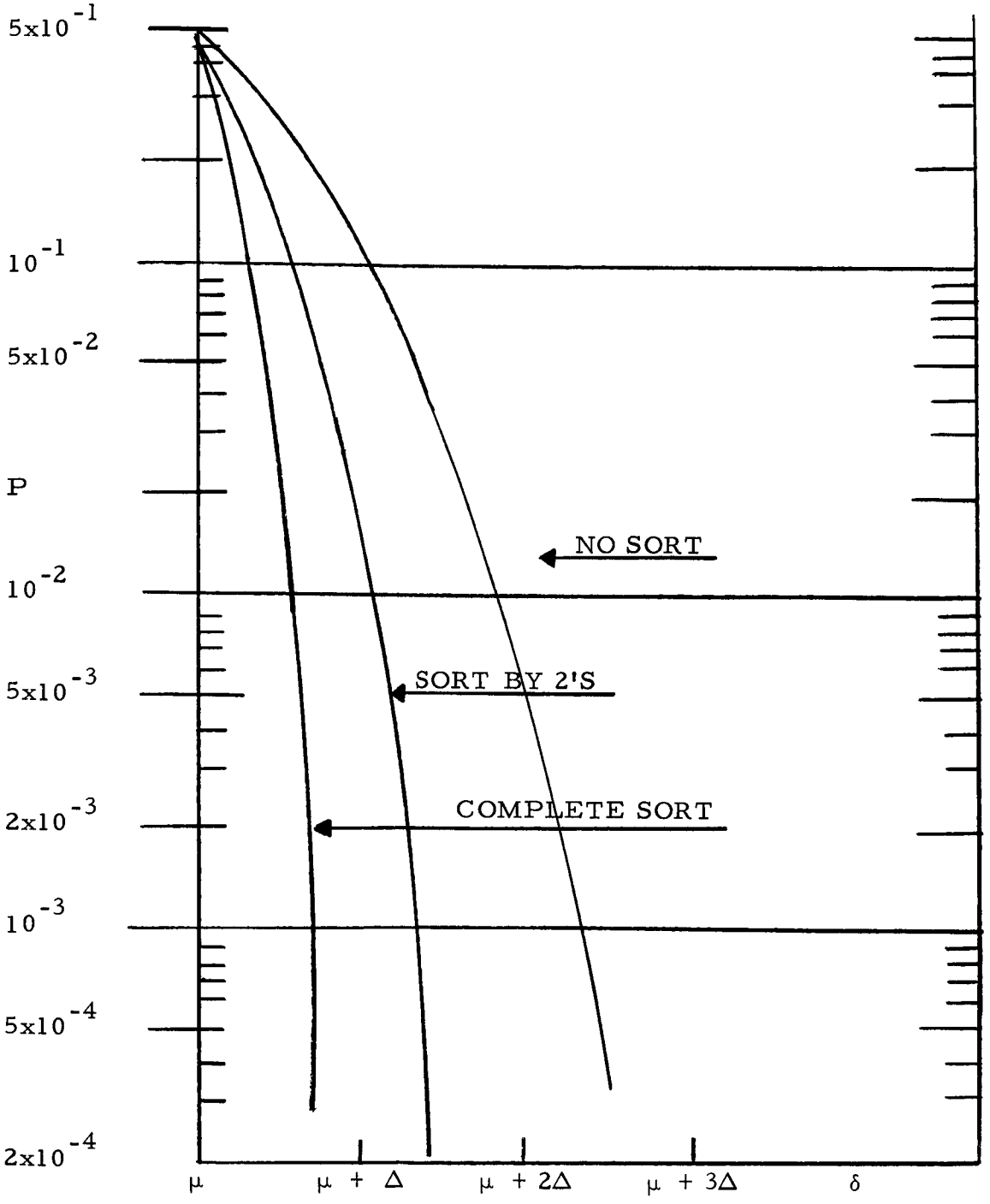


Figure 2.7 Probability (P) That the Chain Delay Exceeds the Ordinate (Triangular Density). Length of Chain = 4 Monte Carlo Simulation 10, 000 Trials.

different statistics. As a case in point, the rippling of a counter is recognized by a chain of combinatorial logic. Here the delay from input pulse to output recognition symbol is the sum of the delays introduced by different types of circuits.

If we adopt the variance of the delay density function as a measure of the spread of the delays from the mean and if the elements are statistically independent then

$$\sigma_T^2 = \sum_{j=1}^{M} \sigma_j^2$$

σ_T^2 = variance of path delay

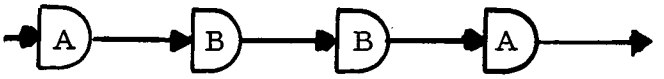
σ_j^2 = variance of individual delays

If one of the elements has a variance much larger than the others, it will dominate the result and any sorting involving the other components will have a negligible effect.

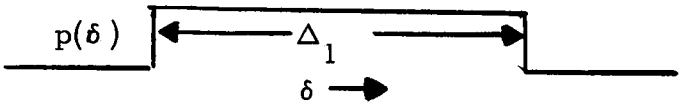
If instead of the variance, the range of possible delay in the chain is used as a measure of the spread in delay, a similar result follows since the range of the total path delay is the sum of the ranges of the individual path delays.

The situation is illustrated by an example shown in Figure 2.8. There element type A dominates the total path and any sorting done on element type B has a negligible effect on the total delay statistics.

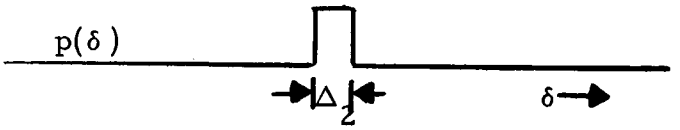
Considerable effort could, at this point, be expended in examining different types of probability density functions and developing approximate rules for desireability of sorting among components with different delay statistics. However, as the next part indicates, a Monte Carlo type simulation appears desireable for the evaluation of the statistical design. With a Monte Carlo simulation, each case of chains of elements with different statistics can be evaluated on its own merits with relative ease.



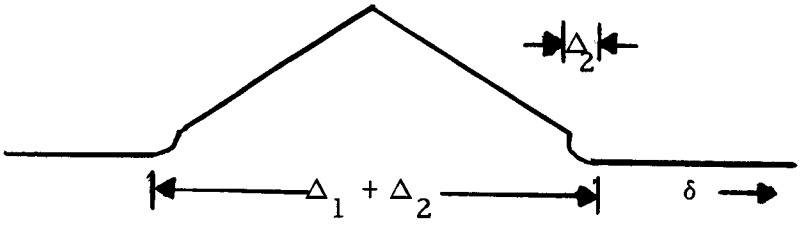
(a) A Hypothetical Chain



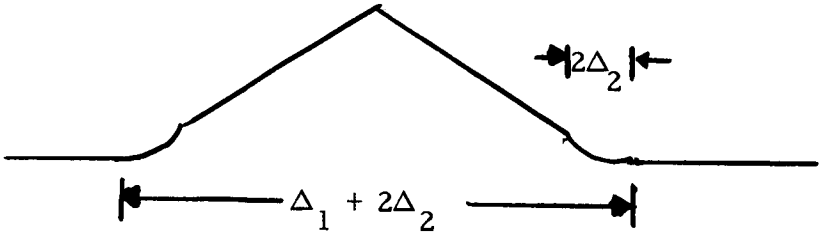
(b) Type a Element Delay Density



(c) Type B Element Delay Density



(d) Results of Sorting Both A and B Elements



(e) Results of Sorting Only A Elements

Figure 2.8 The Effect of a Large Disparity in the Range of Delays for Different Elements.

CHAPTER III

EVALUATION TECHNIQUES

3.1 Linear Techniques

There are several techniques for obtaining an evaluation of the statistical behaviour of a system. Objections can be found to each of them, however, and the problem is to select the optimum technique.

The first technique is the use of the central limit theorem which states that the limiting form of the probability density function of the chain of delays will approach the gaussian density function. The mean of the gaussian function is the sum of the means of the individual element delay density functions and its variance is likewise the sum of the individual variances. This technique has the advantage of simplicity. However, there are distinct disadvantages:

(1) The validity of the gaussian limit is not very good for short chains. The central limit theorem applies as the number of links in the chain is made very large.

(2) The validity of the gaussian density function decreases as one approaches the skirts of the density function.

(3) Since the theorem is based on the use of sum of weighted independent random variables, there is an implicit requirement for a unique propagation path.

(4) There is no provision in this model for the case of a missed path, that is, a signal which has a finite probability of missing a link. An example would be the arrival of a transient at an AND gate after its enabling signal has departed.

A very pertinent example of the use of this technique is in PERT analysis, where the "propagation delays" are analogous to the project durations. In the standard usage of PERT, the project durations are each presumed to have a beta distribution whose statistics can be obtained from the customary "pessimistic, average, and optimistic" estimates of the project duration. Since in a PERT analysis a job completed is never undone, the fourth objection does not apply. Van Slyke[9] has provided an excellent

analysis of the effect of other objections on the validity of the results. Figure 3.1 in particular points out the effect of multiple critical paths on the resultant delay distribution function.

The real significance of objection number three is perhaps best illustrated by a specific example. Consider the AND gate of Figure 3.2. This gate has two inputs with the given density functions of arrival times of on-going transients. Here they overlap, and one input does not uniquely determine the initiation of the propagation delay, since it is the last positive transient which causes the signal to propagate through the gate.

The probability that the propagation delay is initiated at time t is given by:

$$p(t) = p_A(t)\, p(b < t) + p_B(t)\, p(a < t) \qquad (3.1)$$

where $p_X(t)$ is the probability that input X has a transient arrive at time t.

$p(X < t)$ is the probability that the transient on input X has occurred prior to time t.

Note that:

$$p(X < t) = \int_{-\infty}^{t} p_X(t)\, dt. \qquad (3.2)$$

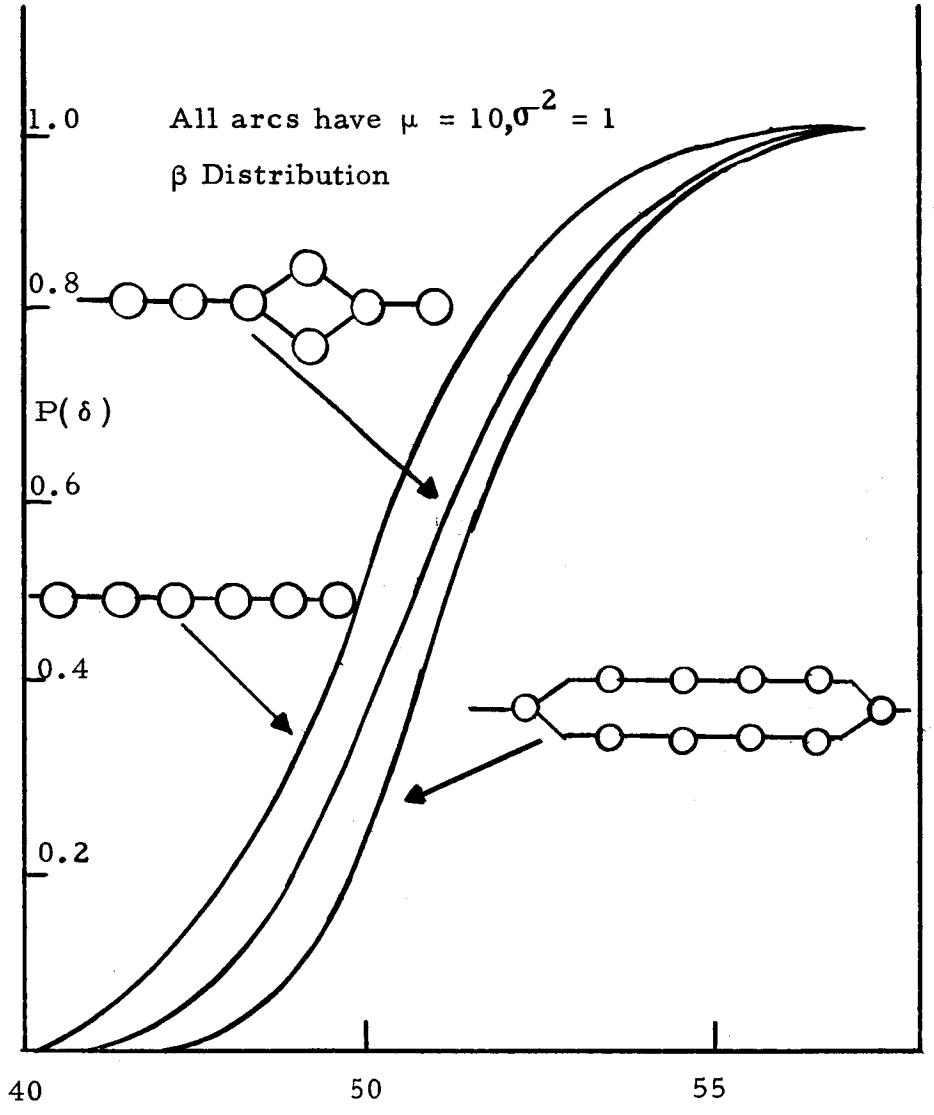
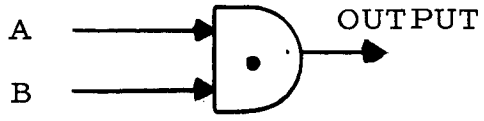
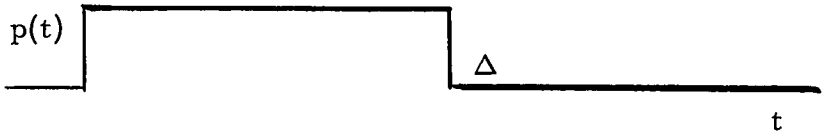


Figure 3.1 Correlation Between Parallel Paths (From "Monte Carlo Methods and the PERT Problem" by R. M. Van Slyke. ORC 63 - 11(RR) Operations Research Center, University of California, Berkeley, May, 1963.)

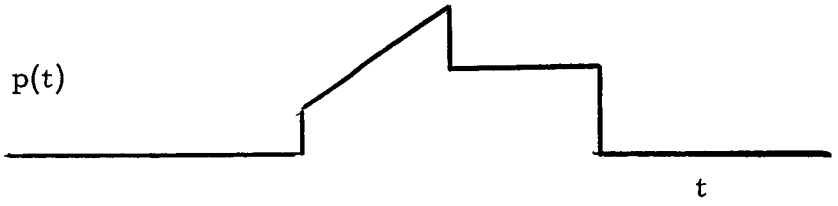


(a) Two Input and Gate



(b) Arrival Time Density of Input B

(c) Arrival Time Density of Input A



(d) Time of Initiation of Propagation Density Function

Figure 3.2 Non-Linear Effects at AND Gates.

For the example shown then

$$p(t) = 0 \qquad t < T \qquad (3.3)$$

$$= \frac{t}{\Delta^2} + \frac{1}{\Delta^2}(t - T) \quad T < t < \Delta$$

$$= 1/\Delta \qquad \Delta < t < \Delta + T$$

$$= 0 \qquad \Delta + T < t$$

It is this density function which must be convolved the propagation delay of the AND gate in question. It is clearly not a linear function of the input arrival time density.

By extension of the reasoning contained above to the case of a flip-flop, it can be shown that the requirement for linearity eliminates from consideration conditions of "races" or "hazards," which are often the prime considerations in the design of a digital system.

The results of equation (3.1) can easily be generalized for an N input gate into the form:

$$p(t) = \sum_{j=1}^{N} p_j(t) \prod_{j \neq i} p(i < t) \qquad (3.4)$$

A widely used statistical technique which has the advantage of overcoming the first two objections to the central limit theorem application is the actual convolution of the density functions involved in the chain. This is most expeditiously done on a digital computer by the use of the impulse convolution technique in those cases where a piecewise linear approximation is applicable to the density functions in the chain. It is described in some detail in Appendix A and can be used in those cases where a unique critical path is certain and where its increased accuracy at the skirts of the resultant density function is required.

3.2 Monte Carlo Technique

A third alternative is to use the Monte Carlo technique. In this method, the system is actually modeled a large number of times. On each modeling, the parameter of interest (delay in our case) is modified according to the statistics which govern that parameter. The Monte Carlo technique overcomes most of the previous objections since it makes no requirement of linearity. It does require a large number of modelings in order to obtain statistics of reasonable validity. This requirement can be translated into a requirement for a high speed digital computer for use in the simulations. A discussion of the number of modelings

required is contained in the paper by Van Slyke[9].

A more serious objection to the use of Monte Carlo techniques is that the data derived can not be verified.* The technique is based on random number generating algorithms which are in actuality pseudo random number generators. One such pseudo random number generator is described in Appendix C. There are numerous instances of apparently excellent random number generators becoming pathological on closer examination. Lynch[10] relates the example of a generator in use for several years which was subjected to the following test. Numbers were drawn in pairs and used to form the x, y coordinates of a unit square. Instead of the "random noise" pattern one would hope for, there existed stripes where no points ever occurred. The point here is not that the algorithm was bad; in many applications it was clearly excellent. Nor is the point that the algorithm was not thoroughly tested. These algorithms cannot be completely random; they are pseudo random and limited in randomness by the finite size of computer word. Many times the non randomness is hidden until a particular case arises. An example of this non randomness is also discussed in Appendix C.

---

*Data cannot be checked always easily.

Of all the evaluation techniques discussed the one judged most suitable by the writer for use by digital systems engineers was the Monte Carlo simulation. It was felt that the risks inherent in the pseudo random number generator were more than compensated by the ability to handle the non linear cases of interest in digital systems. The risk in the use of the pseudo random generation of numbers can be minimized in two ways:

(a) Any serial correlation effects (see Appendix C for example) can be reduced by a semi-random ordering of the random number selection. That is, generate a group of numbers by an algorithm; then select by a random scan of the group.

(b) If anomolies occur in the data outputed, an alternate generating algorithm could be provided. This may have its own pathology, but it is hoped that the pathology of the two generating algorithms is not common.

With this in mind a simulation routine was written for the DDP-116 digital computer. The DDP-116 is high speed (3.4 μ sec add time) digital computer, and its programming is described in the next two chapters.

CHAPTER IV

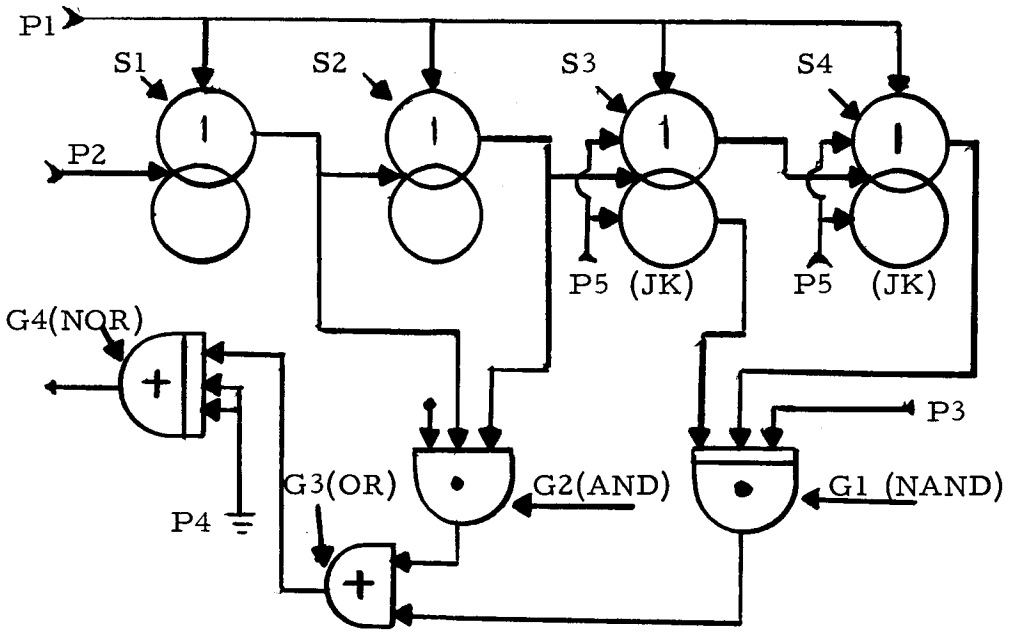
THE SIMULATION PROGRAM

4.1 General Properties

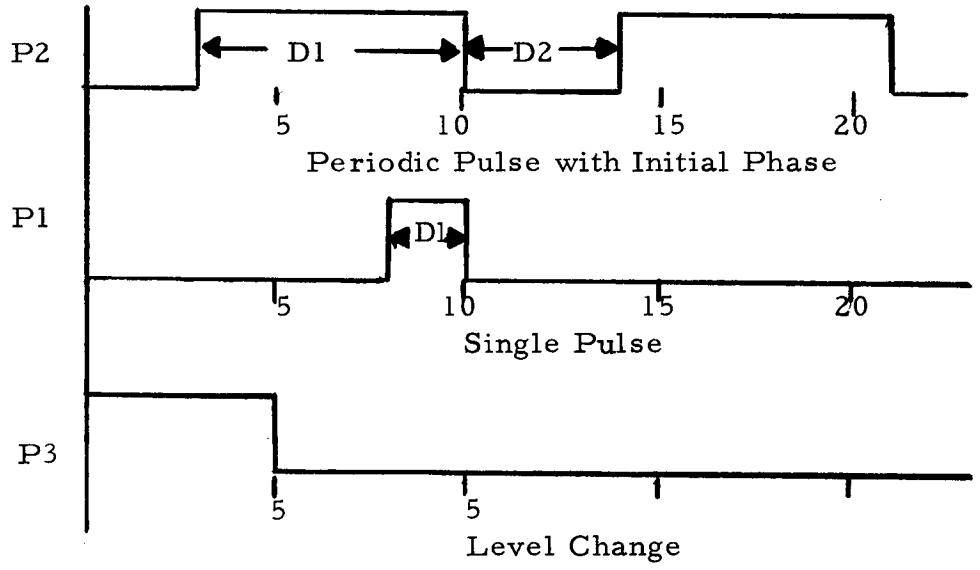
Once the decision was made to perform the statistical analyses of digital systems by Monte Carlo techniques, it was necessary to design a computer program to this purpose. It was determined to make the program as general as possible, so that the result would be a general purpose simulator which could be adapted easily to statistical analysis, rather than a program specifically designed for Monte Carlo analysis. It was hoped that a generally oriented main program could be obtained, with the statistical operations imposed by selected subroutines.

A second goal was simplicity for the user. No knowledge of the computer hardware or software is assumed, the user is presumed only to be familiar with the logic diagrams of the system he desires to analyze, and the statistics he is applying. It is felt that this approach will result in a more useful engineering tool.

The requirements imposed on the user are first illustrated by an example. Suppose that the circuit of Figure 4.1



(a) Circuit Logic Diagram



(b) Waveshapes for Input Signals

Figure 4.1 Circuit Example

is to be analyzed. It should be noted that this circuit is not an optimum circuit, but is utilized solely to permit discussions of all card formats.

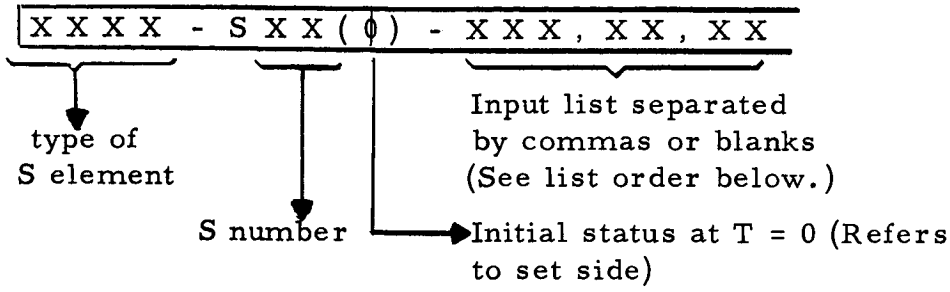
The user first inspects his logic diagram and assigns numbers to his circuit units according to the following scheme:

(a) S numbers are assigned sequentially from one to all storage elements.

(b) G numbers are assigned sequentially from one to all combinatorial elements.

(c) P numbers are assigned sequentially from one to all signals arising external to the system under test.

The user then prepares one IBM care for each circuit unit according to the formats shown in Figures 4.2 to 4.4 along with specific examples of the cards drawn from the circuit of Figure 4.1. The input list referenced in the G and S card formats is merely that: a list of the circuit units providing inputs to the element on the card. These inputs are specified by the G, S, or P number of the system unit serving as input, with the addition in



INPUT LIST ORDERING

ENTRY	STORAGE TYPE	
	RST	JK
1	R INPUT	J INPUT
2	S INPUT	K INPUT
3	T INPUT	C INPUT
4	-	PJ INPUT
5	-	PK INPUT

NOTE The number of entries in the input list must be 3 for an RST element, 5 for a JK element.

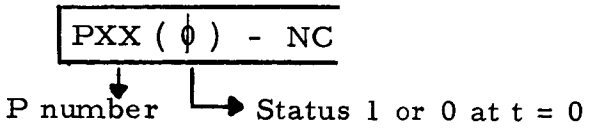
S CARDS FOR THE S ELEMENTS OF FIGURE 4.1 ARE:

```
FFRS-S1(1)-NC,P1,P2

FFRS-S2(1)-NC,P1,S1S

FFJK-S3(0)-P5,P5,S2S,P1,NC

FFJK-S4(0)-P5,P5,S3S,P1,NC
```

Figure 4.2. S Card Format with Examples from Figure 4.1

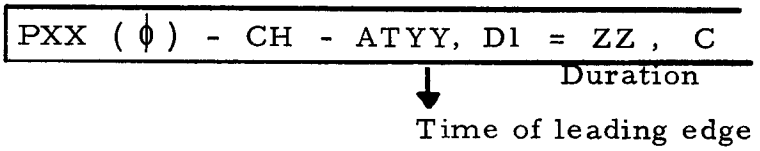
P CARD FORMATS

(a) STEADY LEVEL

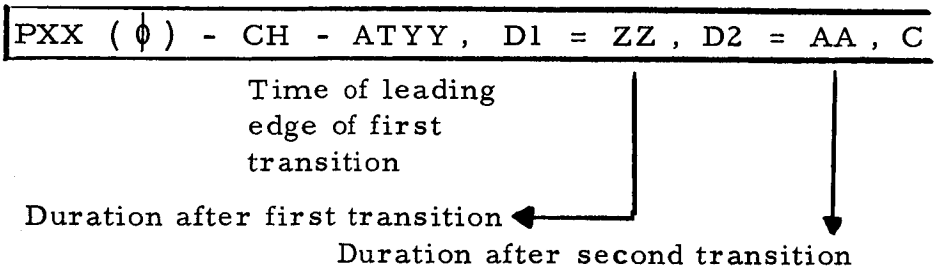


(b) LEVEL CHANGE AT TIME YY

(c) SINGLE PULSE LEADING EDGE AT YY, DURATION ZZ



(d) PERIODIC PULSE TRAIN FIRST TRANSITION AT YY, DURATION OF FIRST TRANSITION ZZ, AND DURATION OF SECOND TRANSITION AA

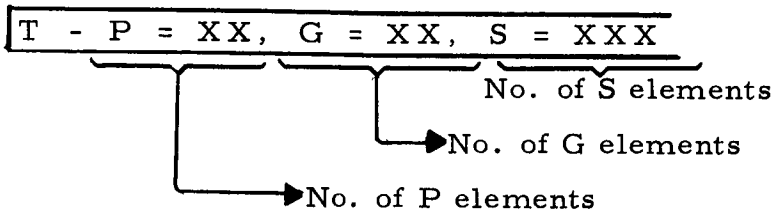


P CARDS FOR THE INPUTS OF THE CIRCUIT OF FIGURE 4.1 WOULD BE

P1 (9)-CH-AT8, D1 = 2, C
P2 (0)-CH-AT3, D1 = 7, D2 = 4, C
P3 (1)-CH-AT5, C
P4 (0)-NC P5 (1)-NC

Figure 4.3 P Card Format with Examples from Figure 4.1.

T CARD FORMAT

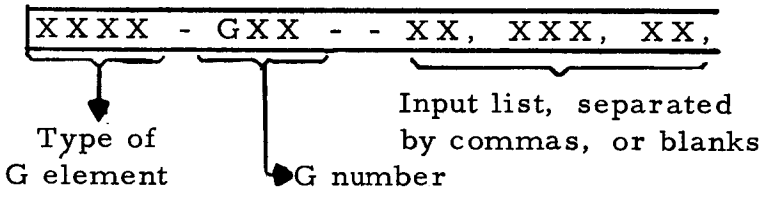


- represents a blank column

THE T CARD FOR THE CIRCUIT OF FIGURE 4.1 is:

T - P = 4, G = 4, S = 4

G CARD FORMAT



NOTE - THE NUMBER OF ENTRIES IN THE INPUT LIST IS EQUAL TO THE NUMBER OF INPUTS SPECIFIED IN THE GATE TYPE.

G CARDS FOR THE G ELEMENTS OF FIGURE 4.1 are:

NAND3-G1--P3, S3R, S4S

AND3 - G2--S1S, NC, S2S

OR2 - G3 --G1, G2

NOR3 - G4 --P4, P4, G3

Figure 4.4 T and G card Formats with Examples from Figure 4.1.

the case of S elements that a suffix R or S is added to indicate that the input comes from the Reset or Set output of the storage element. In the case where an input is unconnected an NC entry is required.

The order of the input list is unimportant in the case of G cards; however, in the case of S cards the ordering shown in Figure 4.2 must be followed.

The P cards are capable of describing various input signals described in Figure 4.1.

The computer program developed for this thesis will handle all of the types of circuit units shown in Figure 4.1. This is not an ultimate limitation, merely a limitation imposed by convenience; additional circuit unit types (one shot multivibrators, gated pulse generators, etc.) can be handled with suitable minor modifications of the main program as noted in Appendix D.

Upon completion of the S, G, and P cards, a T card is prepared according to the format shown in Figure 4.4. This card serves as a list of the number of the various types of circuit element cards and is included primarily to avoid an additional pass of the other cards.

The user then prepares a set of D cards which contain the statistical information for the circuit elements' variation in propagation time. The formats for typical types of delay statistics are shown in Figure 4.5. One card is prepared for each element type. In the case of storage element types, the delay may vary according to which input is excited. If this is the case a separate card for each input can be prepared. This latter case is the one illustrated in Figure 4.5. Note that the formats on the D cards depend on the nature of the statistics assumed.

When the cards are all prepared they are placed in the card reader in the following order:

(a) T Card

(b) P Cards, sequentially by P number

(c) G Cards, sequentially by G number

(d) S Cards, sequentially by S number

(e) D Cards, no ordering required.

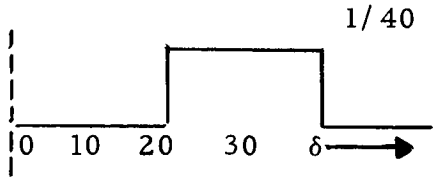
(f) a card with an E in column 1, typically END.

The user then loads into the machine the following tapes, in the given order:

(a) The main program, i.e., the simulator,

D CARD FORMATS

(a) UNIFORM DISTRIBUTION



1) Gate Card

D - XXXX - G - BXXX, WXXX

Gate type

Shortest Delay

Width of Delay

2) Storage Card

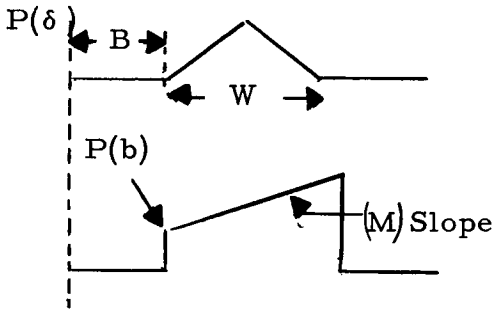
D - XXXX - X - BXXX, WXXX

Input number (See table with S card format)

(b) TRIANGULAR DISTRIBUTION

Same as Uniform.

(c) SLOPED DISTRIBUTION



1) Gate card and storage card same as above through BXXXX. Gate card only shown

D - XXXX - G - BXXX, PBXXX, 1/MXX

P(B)

Reciprocal Slope

TYPICAL D CARDS

```
D - NAND3 - G - B20, W20
D - FFRS - 2 - B10, W25
D - OR2 - G - B5, PB1/64, 1/M512
D - FFJK - 5 - B2, PB1/4, 1/M256
```

Figure 4.5 D Card Formats for Typical Density Functions

(b) The Delay Package, subroutines corresponding to the assumed statistics of the elements,

(c) The subroutine, or subroutines corresponding to the type of analysis desired.

All that remains for the user to do is to press the start button and to enter any requested information at the ASR-33 keyboard.

4.2 The Interconnection Matrix and Delay Table

The information contained on G and S cards is used to generate an interconnection matrix within the computer. The P cards are used to generate pseudo matrix elements and the D cards are used to set up a delay table within the computer.

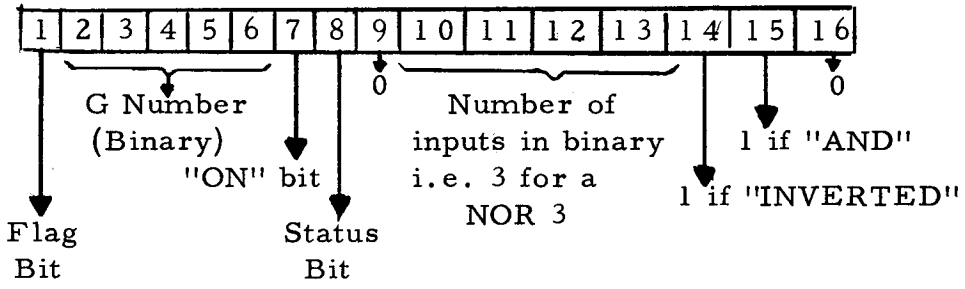
The interconnection matrix consists of a fixed number of 16 bit computer words in each matrix line. The first word is a label word, the second a clock word, and the remaining words (the interconnection words) form a list of all possible input signals. In the interconnection words following the clock word, the bits from left to right represent the P inputs in sequence, then immediately following, the G bits in sequence. When these are exhausted, the bits taken two at a time represent the S elements in

sequence. Two bits are used for each S element since in general each S element provides two outputs. By convention, the first bit in the sequence of two represents the reset side, the second the set side.

One matrix line is required for each G card. The format for the label word is shown in Figure 4.6 and typical examples are shown in Figure 4.10. As the input list of the G cards is scanned by the computer, a one is entered into the bit position in the interconnection words specified by that input. Complete G card matrix lines are illustrated in Figure 4.9.

One matrix line is allotted each connected input in the S card processing. There is then only one entry among the interconnection words for each line. The label word format for the S cards is shown in Figure 4.7 along with some examples in Figure 4.10. Each line label corresponding to the same S card is identical except that the input bit is changed to indicate which input is represented by the matrix line. Matrix lines for the S cards of the circuit of Figure 4.1 are shown in Figure 4.10.

Each P card is given one matrix line. Previously the P cards were said to give rise to pseudo matrix lines. This is

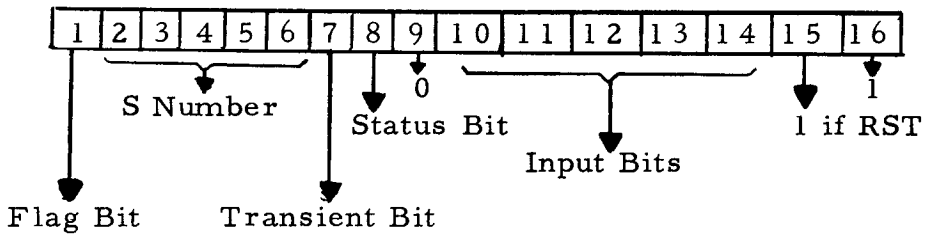


THE STATUS BIT IS 1 IF THE CIRCUIT OUTPUT IS 1. INITIALLY ALL GATES ARE ASSUMED TO HAVE STATUS O.

THE ON BIT IS 1 IF THE COMBINATORIAL FUNCTION OF THE GATE IS ACCOMPLISHED. I.E., ALL INPUTS ARE 1 IN AN "AND" OR "NAND". INITIALLY THE ON BIT AGREES WITH BIT 14.

FLAG BIT IS 1 IF THE CIRCUIT IS TO ENTER DIRECTLY INTO THE STATISTICAL ANALYSIS. INITIALLY IT IS O.

Figure 4.6 G Card Label Word Format



THE STATUS BIT IS 1 IF THE SET SIDE OF THE FLIP-FLOP IS 1. INITIALLY IT IS SET BY THE STATUS INFORMATION ON THE S CARD.

THE TRANSIENT BIT IS 1 IF THE ELEMENT IS UNDERGOING A PROPAGATION DELAY. INITIALLY IT IS 0.

THE INPUT BITS HAVE THE FUNCTION OF IDENTIFYING THE INPUT SPECIFIED BY THE MATRIX LINE.

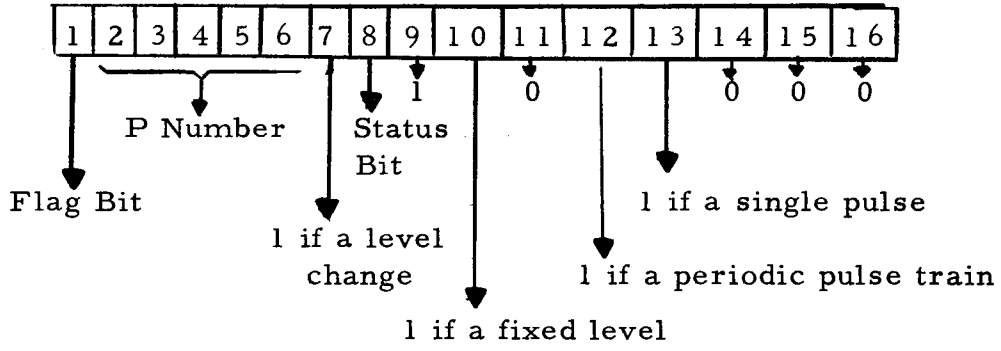
10	11	12	13	14	
0	0	0	0	1	R INPUT IF BIT 15 = 1, J INPUT OTHERWISE
0	0	0	1	0	S INPUT IF BIT 15 = 1, K INPUT OTHERWISE
0	0	1	0	0	T INPUT IF BIT 15 = 1, C INPUT OTHERWISE
0	1	0	0	0	PJ INPUT OTHERWISE
1	0	0	0	0	PK INPUT OTHERWISE

Figure 4.7 S Card Label Word Format

explained by noting that the P matrix lines are treated in the simulation the same as the G and S lines with regard to the clock words (as will be seen later); however, the P cards, representing external signals, have no inputs from within the system, hence no need for interconnection words. The words normally used for interconnection, in a P matrix line are used to store signal shape constants.

Referring to the P card formats in Figure 4.3 for nomenclature, YY is stored initially in the clock word and in the first interconnection word, ZZ in the next, and AA in the last. If the P card in question does not use any or all of these times the corresponding word is left blank. The format for a P card matrix line label word is shown in Figure 4.8, with examples in Figure 4.10, and complete P lines for the circuit of Figure 4.1 are shown in Figure 4.9.

The number of words in a matrix line is obtained by dividing the total number of possible inputs (P + G + 2S) by 16, the number of bits in a computer word, ignoring any remainder and adding three. Since the longest possible P line is 5 words, it is convenient to make this a minimum matrix line length.



THE STATUS BIT IS A 1 IF THE INPUT SIGNAL IN QUESTION IS A 1 AT T = O.

THE FLAG BIT IS A 1 IF THE TRANSITIONS OF THIS SIGNAL ARE REFERENCED IN THE STATISTICAL ANALYSIS.

Figure 4.8 P Card Label Word Format

P1

```
0000010010001000
0000000000001101
0000000000001101
0000000000000010
0000000000000000
```

P2

```
0000100010010000
0000000000000011
0000000000000011
0000000000000111
0000000000001000
```

G1

```
0000011000011110
0001101000000000
0010000000000100
1000000000000000
0000000000000000
```

G4

```
0001001000011100
0000100100000000
0001000100000000
0000000000000000
0000000000000000
```

S1

```
0000010100001011
0000111100000000
1000000000000000
0000000000000000
0000000000000000
0000010100010011
0001001000000000
0100000000000000
0000000000000000
0000000000000000
```

S2

```
0000100100001011
0010011000000000
1000000000000000
0000000000000000
0000000000000000
0000100100010011
0010100000000000
0000000000100000
0000000000000000
0000000000000000
```

Figure 4.9. Typical Matrix Entries for the Circuit of Figure 4.1.

G LABEL WORDS

AND 2 - G4 - - S2S, P1,

0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0

NOR 4 - G12 - - P2, S3R, G8, NC

0	0	1	1	0	0	1	0	0	0	1	0	0	1	0	0

OR3 - G5 - - G4, S 10 S, G2

0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0

NAND3 - G8 - - P1, NC, G14

0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0

S LABEL WORDS

FFRS - S2(1) - S3R, S3S, P1 (R input shown)

0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	1

FFRS - S3(0) - NC, G4, P2 (T input shown)

0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1

FFJK - S4(1) - S1R, S1S, P1, NC, G4 (PJ input shown)

0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1

FFJK - S11(0) - G2, G3, P4, S1R, NC (K input shown)

0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0

P LABEL WORDS

P4 - (1) - NC

0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0

P2 - (0) - CH - AT 20, NC

0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0

P3 - (1) - CH - AT 30, D1 = 40, C

0	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0

P1 - (0) - CH - AT 10, D1 = 20, D2 = 10, C

0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0

Figure 4.10 Label Word Examples.

The D cards are used to create a delay table. This table will vary according to the statistical model used. The case of a uniform density function is treated here as an example.

The table consists of a label word and one or more words of statistical parameters for each line in the table. The label words are independent of the type of statistics used and consist of the last eight bits of the matrix line label, corresponding to an element type, or if desired in the case of storage elements, an element type and input.

The remaining words in the delay table depend on the nature of the statistics. After the label has been generated, control is transferred to the program on the statistical subroutine tape where the remaining words in the line are prepared. In the case of the uniform density of delay, the remainder of the table consists of a single word. The first eight bits are the minimum delay (B) and the last eight bits are the width (W) of the density function.

Figure 4.11 shows the delay table entries corresponding to typical D cards.

LABELS	ENTRIES
0000000000010100	0001010000010100

D - NOR2 - G - B20, W20

0000000000101110	0000111100100011

D - NAND5 - G - B15, W35

0000000000001011	0000101000011001

D - FFRS - 2 - B10, W25 (Set input)

0000000000100001	0001011100001110

D - FFJK - 4 - B23, W14 (PJ input)

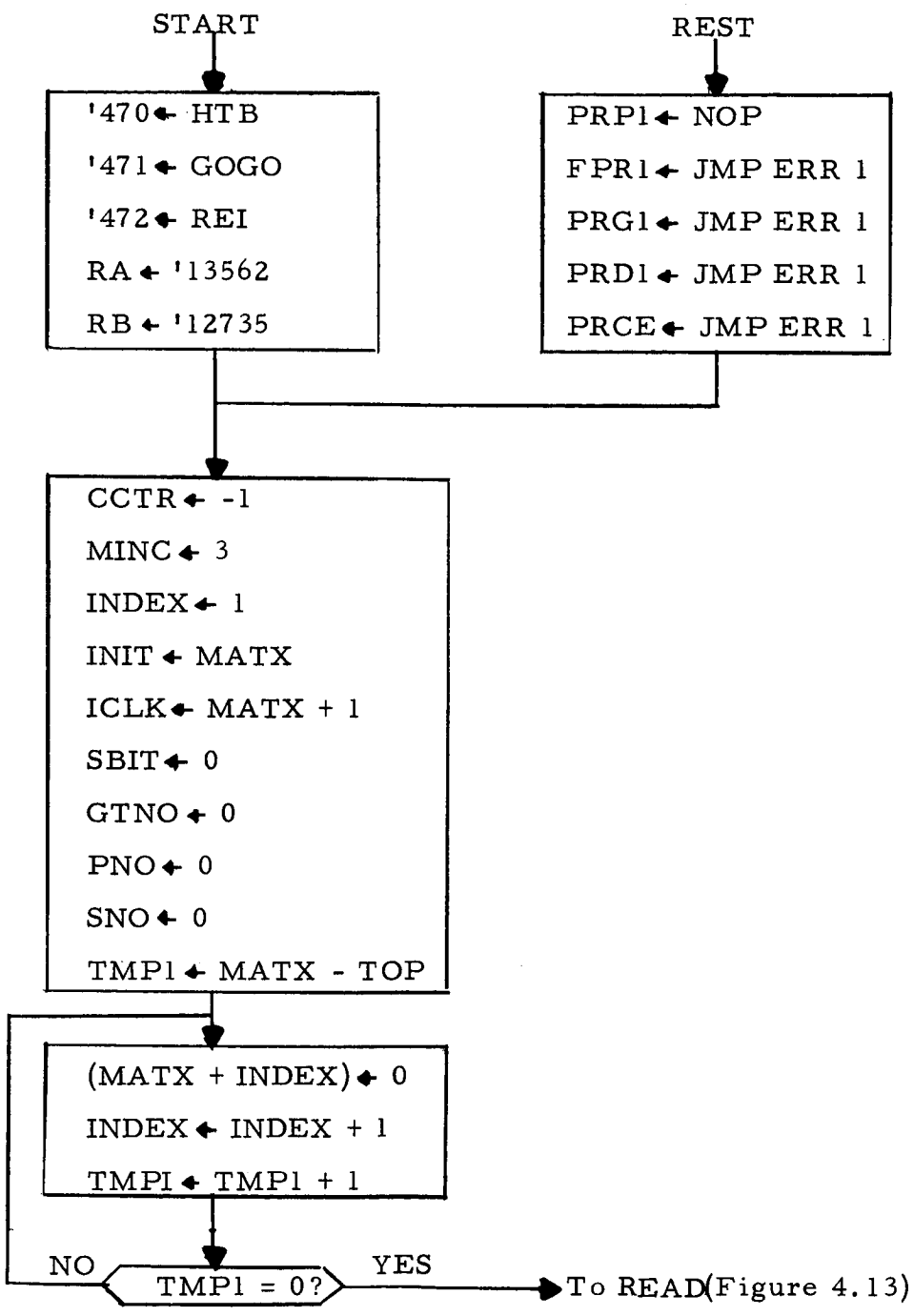
Figure 4.11 Typical Delay Table Entries

4.3 Program Initialization and Card Read Routine

The program initialization routine consists of clearing the computer memory portion to be used for the interconnection matrix and of the establishment of initial values for various constants. It is shown in the form of a flow chart in Figure 4.12.

The interconnection matrix and certain auxilliary registers are located immediately following the main program. A number of words equal to one matrix line between the main program and the start of the interconnection matrix is set aside for storing initial values. These words form two registers INIT and ICLK. Register ICLK is used to indicate which input signals will undergo transients while register INIT is used to store the initial values of the P and S elements. To define these registers, use is made of a base address MATX, a mnemonic given to the first address following the main program. It proves convenient, for reasons to be discussed later, to define the register INIT by the address of the word actually located two words ahead of the actual register.

Also shown on the flow chart is a routine titled REST. This routine is used to restore some initial conditions in event that the card processing is interrupted by a format error. The NOP and JMP ERR1 are DDP-116 instructions for no operation and



NOTE: SEE APPENDIX E FOR FLOW CHART CONVENTIONS

Figure 4.12 Program Initialization Flow Chart

transfer to routine ERR1 respectively. The mnemonics referenced here are latchwords in the card processor which open successively different portions of the processor.

In the portion devoted to the clearance loop, mnemonic TOP refers to the highest allowable memory location for the purposes of forming the interconnection matrix. The statistical and analytical option programs are all stored in memory above TOP.

The mnemonics HTB, REI, and GOGO are the addresses of entry points in the main program which are utilized by the statistical and analytical programs.

The other mnemonic referenced in the initialization flow chart are listed and defined below.

GTNO	is a counter of G cards
SNO	is a counter of S cards
PNO	is a counter of P cards
CCTR	is a total card counter (T, G, P, S, D, and E)
MINC	is the number of computer words in a matrix line
RA	is an initial value of a random number (15 digits)
RB	is an initial value of a random number (remaining 15 digits)

IND is an index register

SBIT is a variable exit word used with subroutine HTBI. Its use is discussed in page 75.

The card reader routine flow chart is shown in Figure 4.13(a). When it is desired to read a card, the card reader is interrogated twice in order to determine if the reader is on and cards are in the hopper, and to determine if a card is registered. If these conditions are not met the program remains in the corresponding interrogation loop. If the conditions are met the card reader buffer is interrogated. If the buffer is not full and ready for transfer, the program waits again in an interrogation loop; if the buffer is ready the contents are read and stored in the program buffer (BUFF). As each card column is read the program buffer location is indexed, so that eventually the entire 80 columns of the IBM card are transferred into BUFF. At this point the index register is re-initialized and the program proceeds to process the information on the card.

During the course of processing, the information from BUFF is transferred to the accumulator at many points on the program. In the interests of brevity (and clarity) the shorthand notation shown in Figure 4.13(b) has been adopted for use in

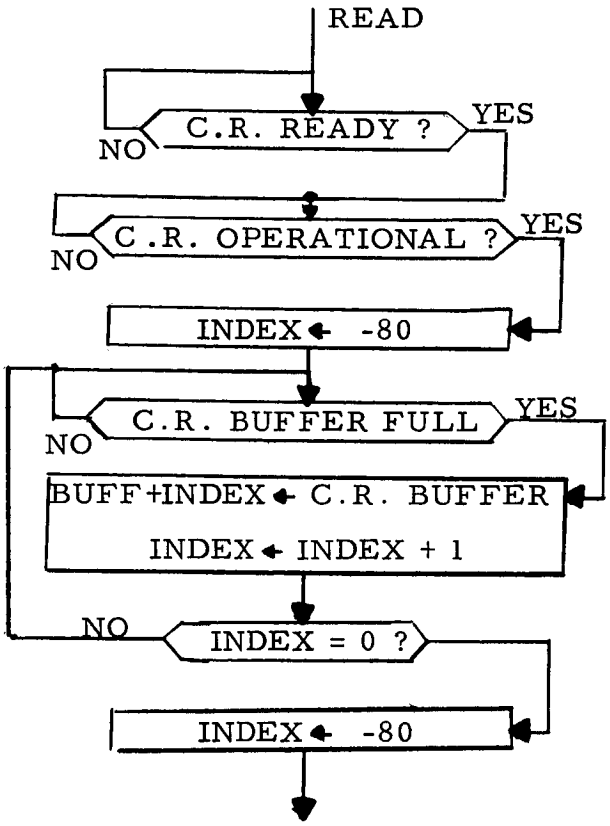


Figure 4.13(a) Card Read Routine

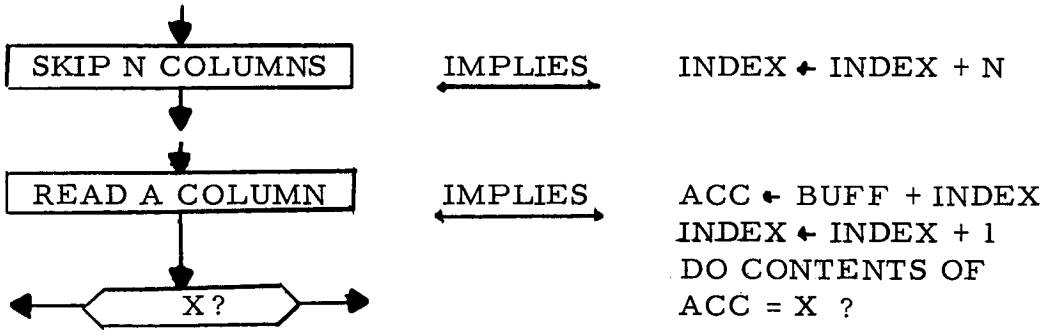


Figure 4.13(b) Shorthand Notation

subsequent flow charts .

4.4 The T Card Processor

The first card read by the program is the T card. The function of the T card processor is to horizontally dimension the interconnection matrix. The flow chart for this processor is shown in Figure 4.14.

As the card is scanned, it is checked for the following types of format errors:

(a) First symbol is not a T

(b) The element type symbols (G, S, and P) are not entered in the proper order. This is done primarily to prevent situations wherein one element type is entered twice and another not at all.

(c) The sequence of symbols following an element type symbol and an equal sign is not a string of numerals terminated by a space or comma. The equality sign itself is not checked, the column after an element type symbol is skipped.

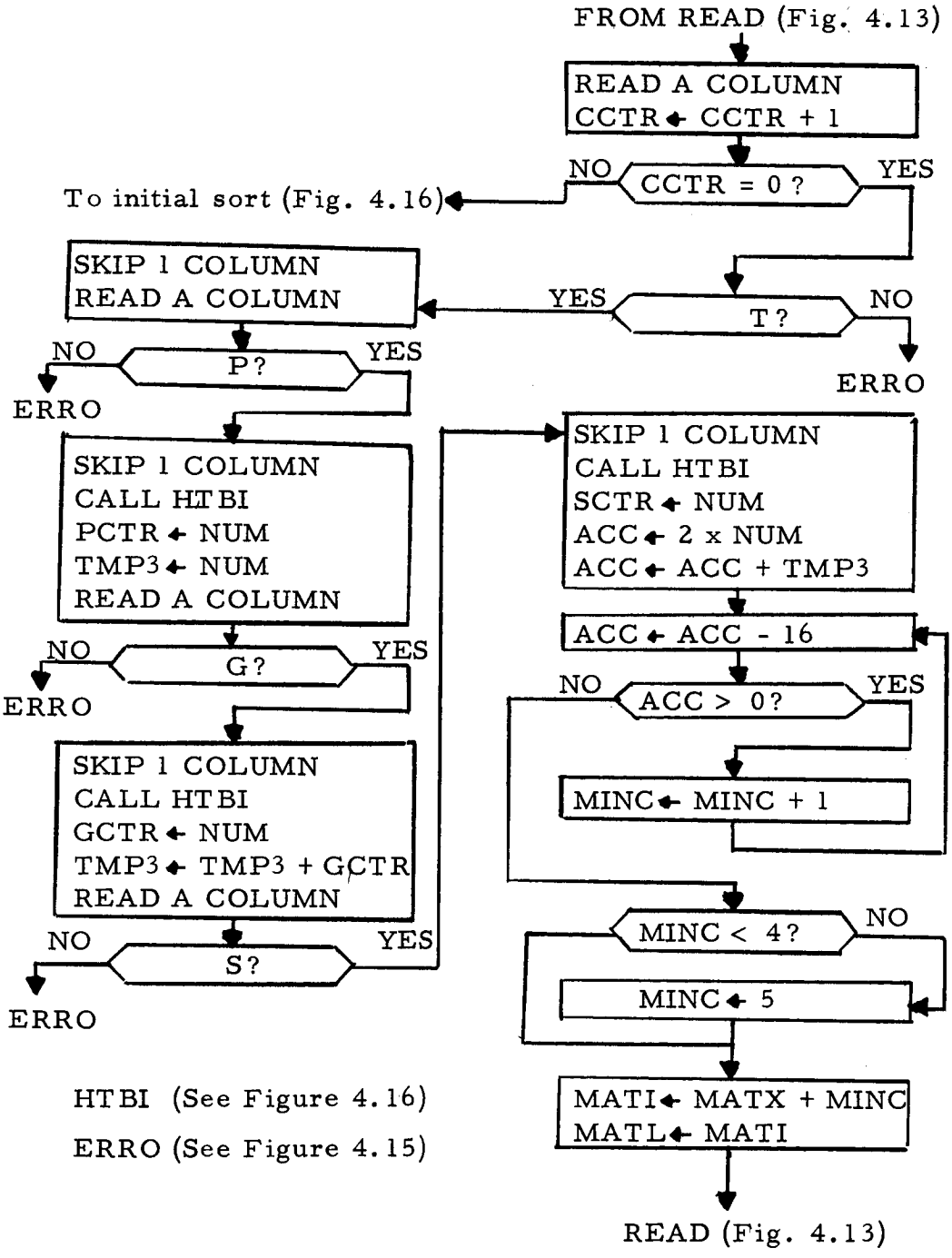


Figure 4.14 T Card Processor Flow Chart

(d) An unallowed symbol appears for an element symbol.

In the event that one of the above format errors occurs, the machine transfers to subroutine ERRO (flow chart, figure 4.15) which causes the typewriter to print out the number of the card and the column being scanned when the error occurs. After the format error is corrected, all cards must be reinserted in the hopper and the start button pushed.

The number of each of the element types is converted from card reader output code to binary and stored for future reference. The total number of possible circuit inputs (P + G + 2S) is computed in TMP1, and converted into an equivalent number of matrix lines. Each time the subtraction of a word length (16) yields a positive result, the contents of MINC are incremented by 1. A negative or zero result from any of the repeated subtraction terminates the process. The length of possible P card pseudo matrix lines dictates that a minimum value of MINC be five. If MINC is less than five, a five is entered into MINC, otherwise the computed value is accepted.

The initial status registers INIT and ICLK are

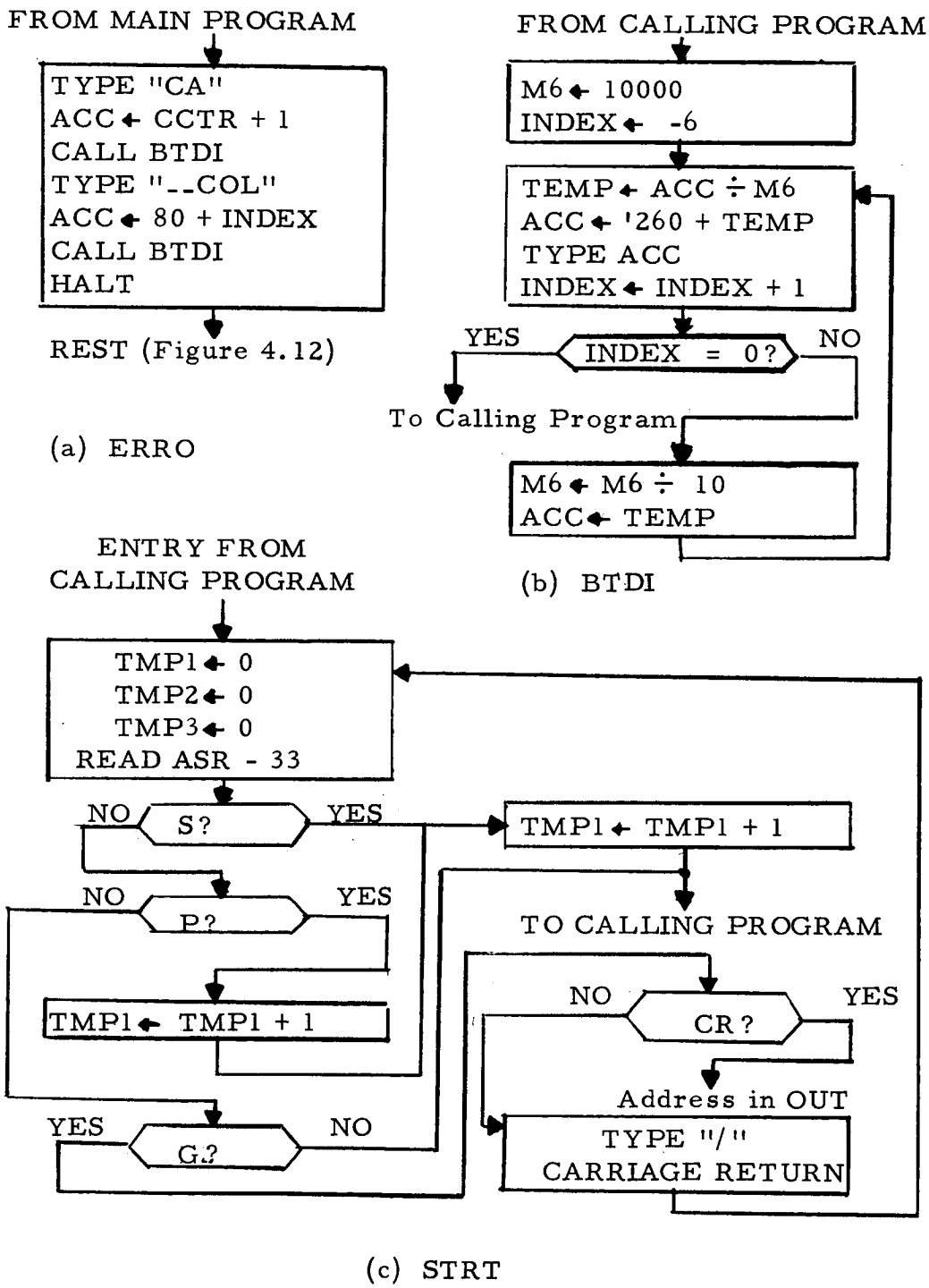


Figure 4.15 Miscellaneous Subroutines

allotted the equivalent of one matrix line immediately preceding the matrix. For this reason the address of the last program entry (MATX) is augmented by MINC and used to define the address of the first word in the matrix (MATI) and also the initial entry into the matrix line index (MATL).

A subroutine called very often in the T card processor and elsewhere in the matrix maker is subroutine HTBI, a "Hollerith to Binary" converter. The flow chart for subroutine HTBI is shown in Figure 4.16. Upon entry into HTBI, the next column on the card being processed is read and checked to see if it is a numeral (input from card reader less than 12 octal). If it is a numeral it is entered in binary into TMP1. The next column is read. If it, too, is a numeral, it is added to previous contents of TMP1 multiplied by 10 and stored in TMP1. The operation is continued until a non-numeral is detected.

If a non-numeral is detected, the variable exit word SBIT is interrogated. If SBIT is zero the only permissable symbols are comma or blank, either one of which results in an exit from HTBI with the binarized word in the accumulator. A non-permitted symbol results in a transfer to ERRO. If SBIT is not zero, only an R or S followed by a comma is permitted. If an

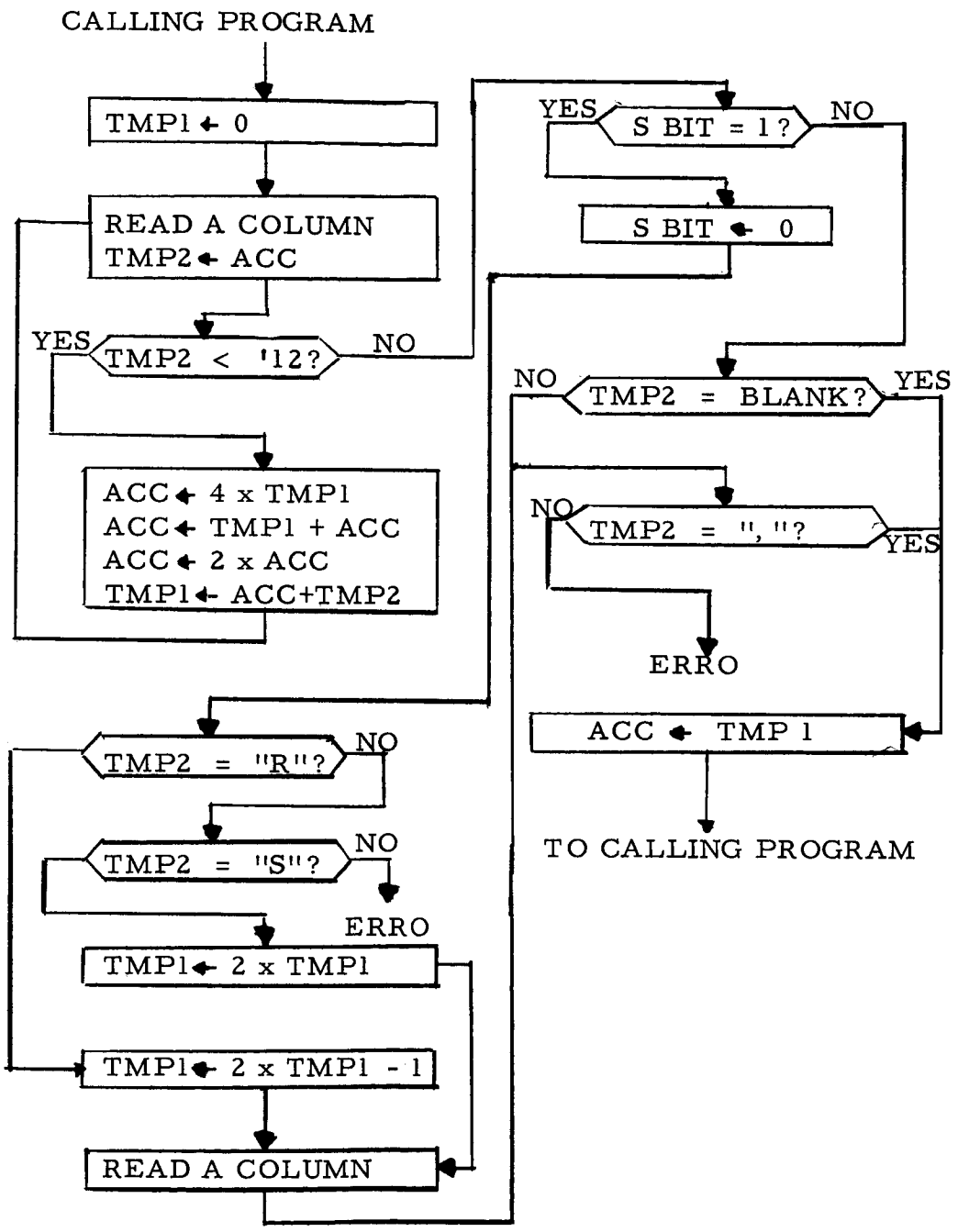


Figure 4.16 Subroutine HTBI, Hollerith to Binary Conversion

S occurs the binarized number is doubled before an exit. If an R occurs the binarized number is doubled and one is subtracted from it before exiting. The reason for the variable exit will be made clear in the discussion of the G card processor. For the case of the T card processor SBIT is always zero and any string of digits in successive card columns terminated by a blank or comma will be binarized and placed in the accumulator by HTBI.

In addition to the mnemonics previously discussed other mnemonics appearing in the T card processor and in HTBI are listed and defined below:

SCTR	number of S elements expected
PCTR	number of P elements expected
GCTR	number of G elements expected
NUM	output of HTBI

When the T card processing is complete, the next card in the card reader is read.

4.5 The Initial Sort and P Card Processor

After the T card has been processed, all subsequent cards are subjected to an initial sort. This initial sort, with the

flow chart of Figure 4.17, has the function of determining which processing routine shall handle the card.

The first column of the card is read and interrogated to see if the symbol F, P, D, or E occurs. If none of these symbols occurs control is transferred to the G card processor where a further sort is made.

If the first card column contains an F, the card represents a storage element. The storage card processor contains a latchword FPR1. This word will normally contain an instruction causing a transfer to subroutine ERR1 which causes the typewriter to print "ER1" and halt operation. However, if, and only if, all P and G cards have been processed, a NOP instruction is inserted into FPR1 which permits further processing of storage cards. If this later case holds, card column three is interrogated for a J or a K. If J occurs the JK element processor is called, If R occurs the RS element processor is called; If neither R nor J occurs subroutine ERR0 is called.

The detection of a P results in transfer to the P card processor. If a D or E is detected control passes to the D or E card processor respectively.

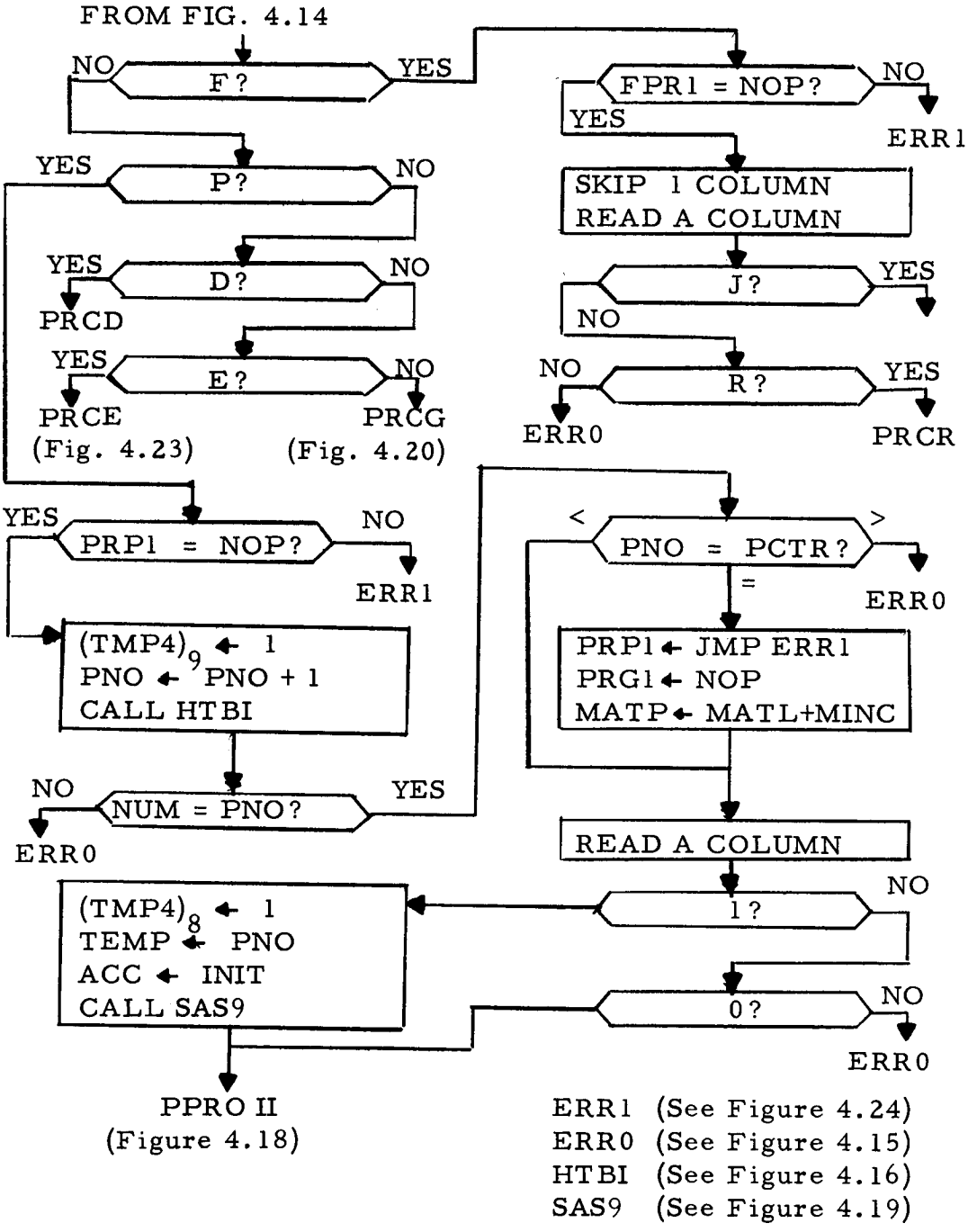
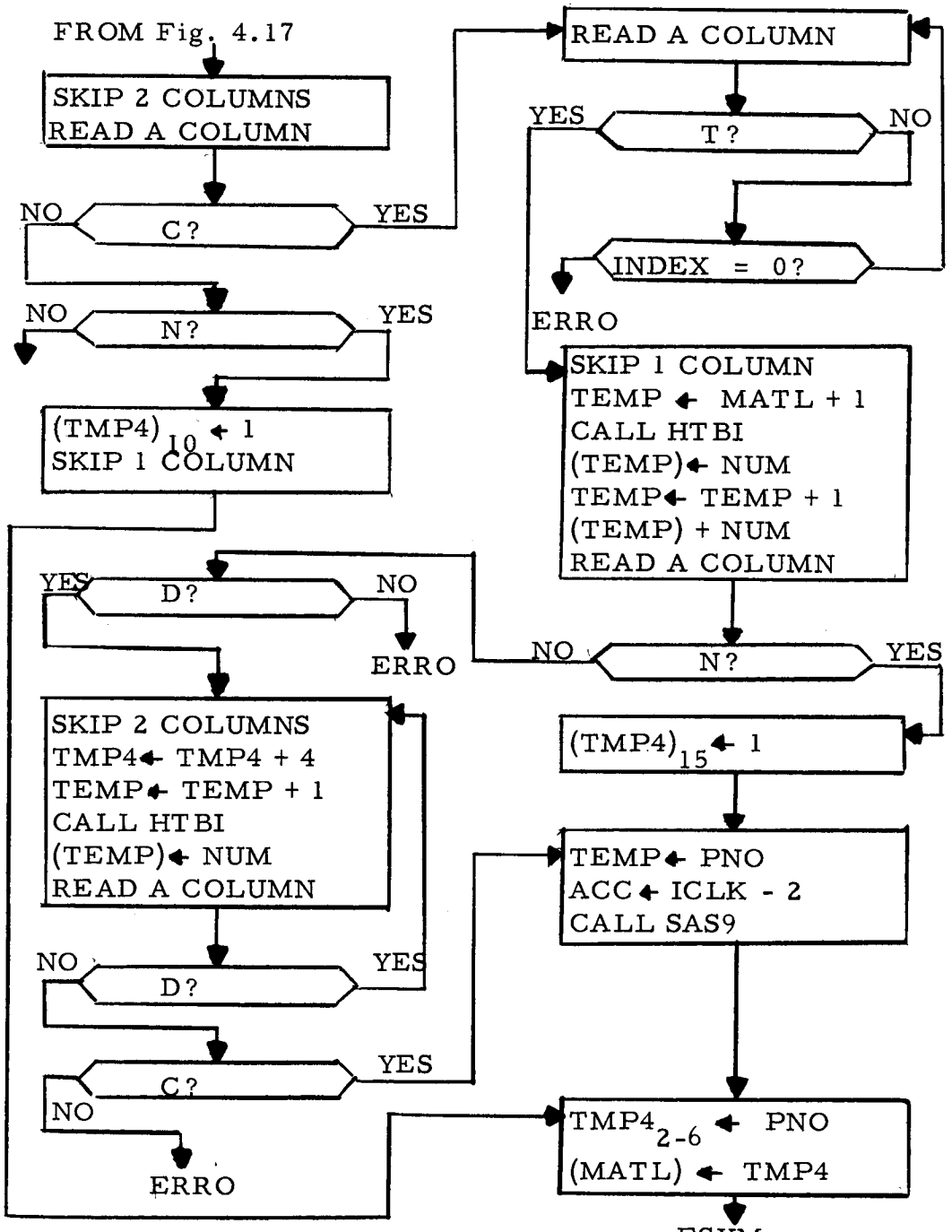


Figure 4.17 Initial Sort and P Card Processor (Part I).

Assuming that the initial sort revealed a P in column 1, the control is transferred to the P card processor. It will be recalled that the information on the P card is transformed into a pseudo matrix line, consisting of a label, a clockword, and up to three words which store signal shape parameters. The flow charts of the P card processor are shown in Figures 4.17 and 4.18. The P card formats were shown in Figure 4.3.

Following the detection of a P in the first card column during the initial sort, the latchword PRP1 is inspected. As the program loaded, a NOP was stored there, and the NOP remains until a number of P cards equal to the number stated on the T card have been processed. At that point a jump to ERR1 is inserted into PRPI. Thus if more P cards than predicted are fed into the machine an error type out will occur.

Presuming that PRPI contains a NOP, the second card column and any succeeding columns containing numerals are read and binarized. This binary number is compared with the contents of the P card counter (PNO). Since PNO is incremented by 1 each time a P card is processed, this test indicates whether the cards are in sequence. A card out of sequence results in a transfer to ERRO. The contents of PNO are then compared with the contents



ESYM (See Figure 4.19)
HTBI (See Figure 4.16)
SAS9 (See Figure 4.19)
ERRO (See Figure 4.15)

Figure 4.18 P Card Processor (Part II)

of PCTR. If these numbers are identical, the P card under process is the last P card expected. The P processor is then latched by setting a JMP ERR1 into PRP1 and the G card processor is unlatched by inserting a NOP into PRG1 the gate processor latch-word. The matrix line index (MATL) is augmented by MINC and stored in MATP where it represents the line index for the last P card.

As long as PNO is less than or equal to PCTR processing continues. The next entry on the card contains the initial level of the signal, enclosed by parentheses. The column containing the left parentheses is skipped and the status column is read. It must contain a 1 or a 0 or a transfer to ERRO is made. If a 1 is present as an initial status, a 1 is entered into a specific bit of the register INIT. The format of the bits in INIT is the same as that of a matrix interconnection word. Thus, if PX is initially a 1, a 1 is entered into the $X^{\underline{th}}$ bit of INIT. If the status bit on the P card is 0 no entry is made into INIT, and the $X^{\underline{th}}$ bit remains a 0.

Entry into INIT is made by subroutine SAS9, shown in Figure 4.19. SAS9 makes use of two pieces of information. The contents of the accumulator are treated as a reference address,

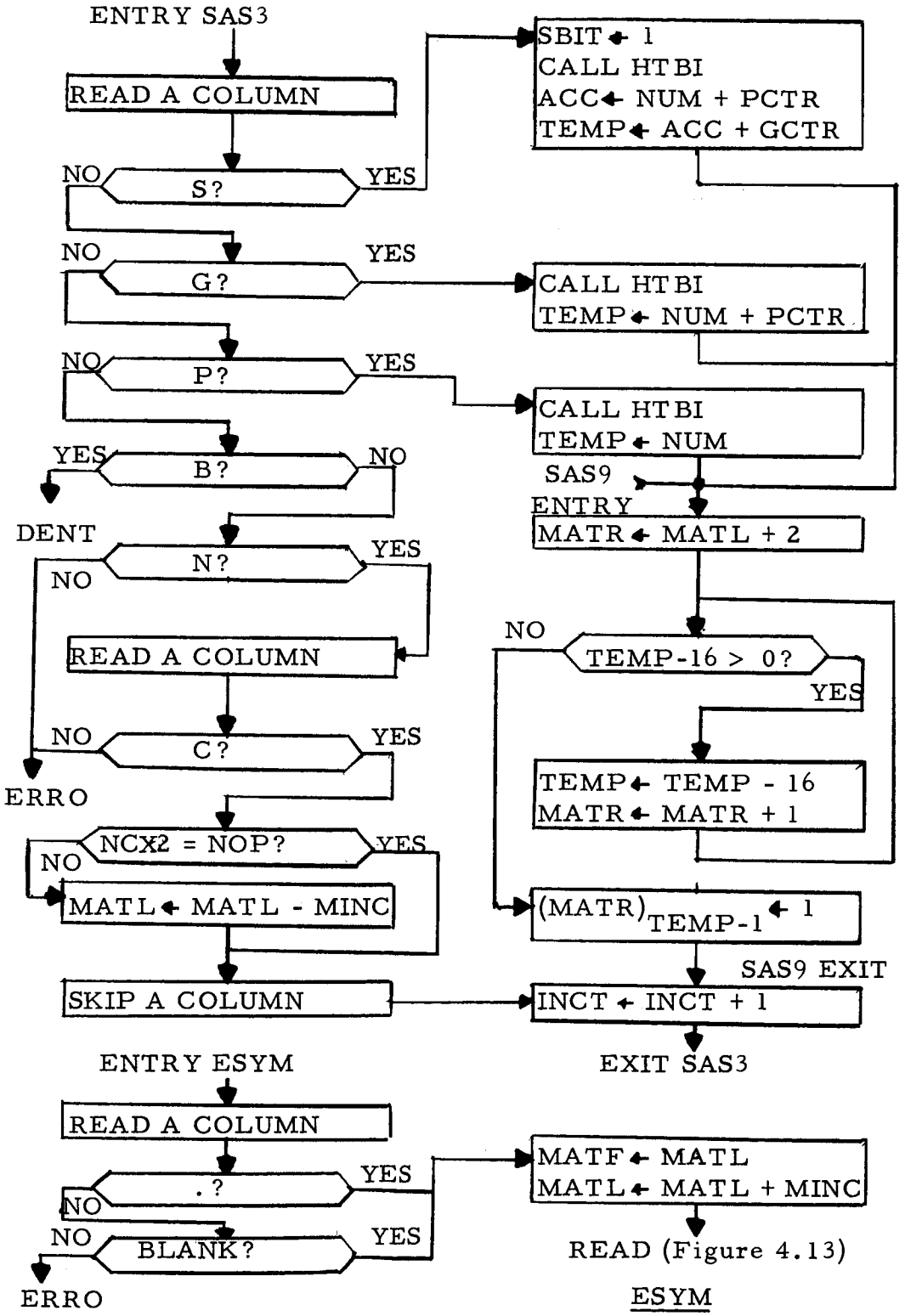


Figure 4.19 SAS9, SAS3, and ESYM

and the contents location TEMP indicate the number of bits to the right of the most significant bit of the address in TEMP into which a 1 is to be entered. For example, if the accumulator contained the address of INIT and the contents of TEMP were 5, a 1 would be inserted in bit 5 of the address stored in INIT. If TEMP had contained a 27, a 1 would be placed in the $11^{\underline{th}}$ bit of the word following the address stored in INIT. It proves convenient in later application of SAS9 to have 2 added to the accumulator contents, so that the address used to specify INIT is actually that of the word two places ahead of the actual location of INIT Thus the definition of INIT specified in the initialization actually makes INIT composed of all the skipped matrix line except the first two words.

After the initial status is processed, two card columns are skipped, then an interrogation for an N or C is made. If the column in question contains neither symbol, a transfer to ERRO is made. The detection of an N is interpreted as specifying a fixed level and a 1 is inserted in label but 10 and the contents of PNO are inserted into lebel bits 2 through 6, and processing is completed by calling subroutine ESYM. ESYM, shown in Figure 4.19, is an end symbol processor. The next card column is

checked for a blank, or a period. If neither occurs ERRO takes control, if either occurs MATL is incremented by MINC and control is transferred to READ.

After the status has been processed, if a C is sensed rather than an N, subsequent columns are scanned until a T is sensed, or the 80 columns of the card are exhausted. The later case results in a transfer to ERRO, the former results in the numerals in the card columns following the T being binarized and inserted into both the clock word and the first storage word. The binarization is governed by HTBI and terminates when a blank or comma is sensed.

At this point, the next column is interrogated for either an N or a D If neither occurs, a transfer to ERRO is made. Should an N occur, the card represents a level change and a 1 is inserted into label bit 15. A 1 is also inserted into register ICLK. Register ICLK, it will be recalled, is located between the end of the program and the beginning of the matrix. Its use is to indicate which of the external inputs have a future transient on them, as opposed to fixed levels. ICLK is located immediately ahead of INIT and can occupy at most two words, restricting the operation to a maximum of 32 external inputs. This limit on the size of ICLK

could be removed by allowing two matrix lines between the end of the program and the start of the matrix; it was not considered necessary, however, at this time.

SAS9 is used to load ICLK. The contents of PNO are placed in TEMP and the address stored in ICLK (decremented by 2) is placed in the accumulator. After ICLK is loaded, the contents of PNO are placed in label bits 2 through 6, and an exit to ESYM is made.

If, following the loading of the clock word and the first storage word, the next column had revealed a D rather than an N, the two succeeding columns are skipped and the numerals contained in subsequent columns are binarized by HTBI. The results of the binarization are placed in storage word two and a 1 is added to the label word in bit position 14. If at this point the next column yields a D, the process described in the paragraph is repeated except that the binarized information is placed in storage word three. Note that the double addition of the 1 in bit 14 results in a 1 in bit position 13.

The two cases discussed immediately above correspond to a single pulse (single D on the card) and a periodic pulse (two

D's on the card). Each of the two cases is terminated by the symbol C, which causes an entry to be made in ICLK, and the contents of PNO into label bits 2 through 6, after which an exit to ESYM is made. If the last entry on the card is not a C, an exit to ERRO is made.

4.6 The G Card Processor

The flow chart for the G card processor is shown in Figure 4.20. After the initial sort, if the first card column does not contain an F, P, D, or E entry is made to the G card processor. Here the first column is inspected for an N, A, or an O. If none of these symbols occurs an exit is made to ERRO.

If an N is detected, a 1 is inserted into label bits 7 and 14. Bit 14 is the bit which specifies whether or not an inversion is contained in the gate. And since the initial output of the gate is assumed to be zero, the "ON" bit which specifies the status of the combinatorial portion of the gate is always a 1 if the N bit is a 1. Since the first column symbol N is used as a prefix, after the detection of an N, the second column is interrogated for an A or an O. If an A is detected in the first column or the NA combination in first and second columns a 1 is entered into label list 15.

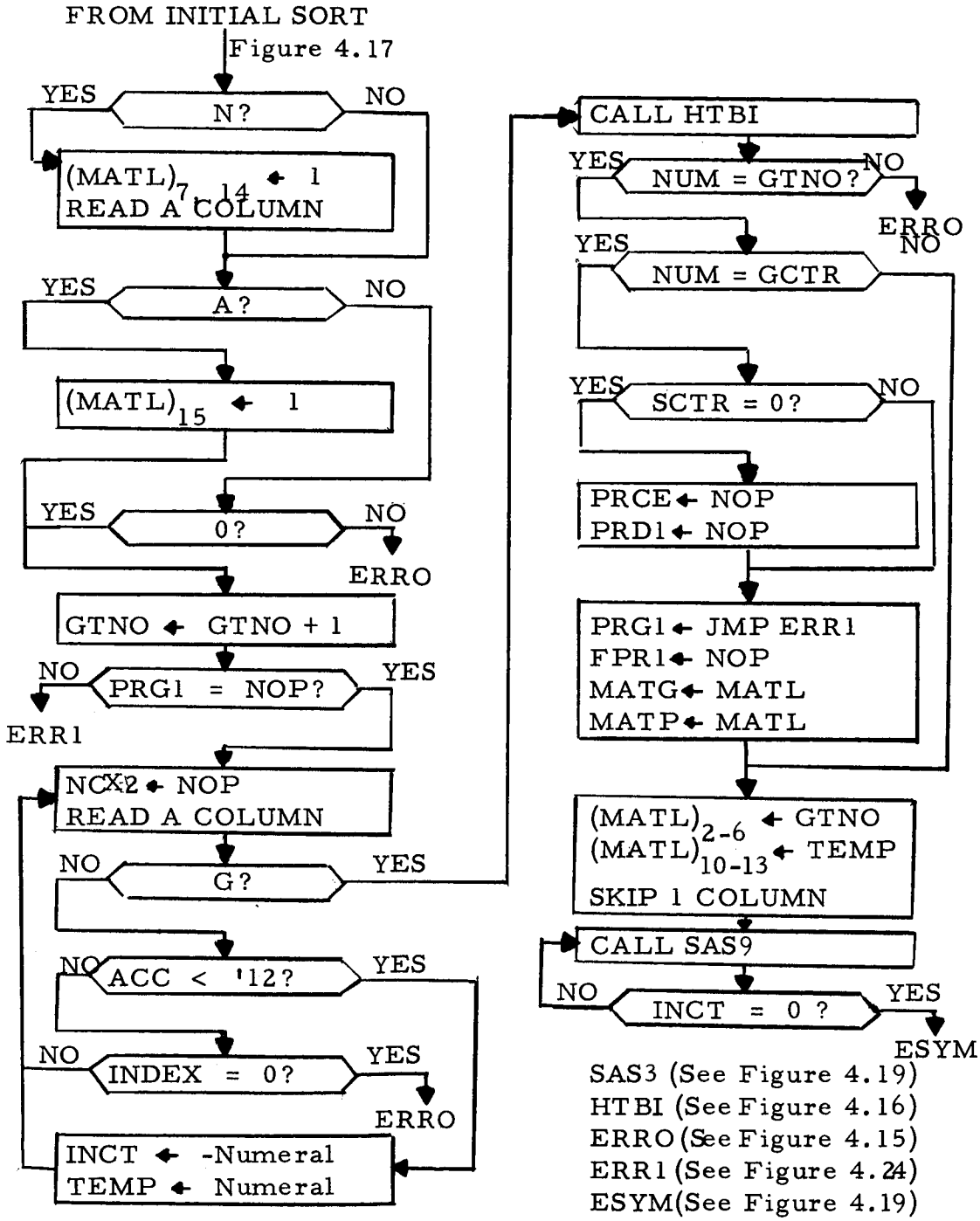


Figure 4.20 G Card Processor

After the inversion and combinatorial information has been entered in the label the gate number (GTNO) is incremented by 1. At this time, the gate processor latch word (PRGI) is interrogated. This word is normally an exit to ERR1. However, as was previously seen, a NOP is inserted into PRG1 upon completion of the P card processing.

At this point, a NOP is inserted into control word NCX2 with an effect to be described in the discussion of input list processing. Card columns are read until a numeral is detected. This numeral represents the number of inputs into the gate; it is negated and inserted into the input counter (INCT) as well as inserted into label bits 10 through 13. The present method of processing restricts the number of inputs to nine or less. A trivial modification involving a call to HTBI could easily eliminate this restriction.

Card columns are scanned until the symbol G is detected or the 80 columns are exhausted. The later case results in a transfer to ERRO, the former case results in the binarization of the numerals following the G symbol by subroutine HTBI. This binarized number is checked against GTNO to see if the cards are in sequence and compared with GCTR to see if it is the last G card.

If the card is out of sequence ERRO is called. If the card is the last G card, the gate processor is latched by inserting a JMP to ERR1 into PRG1 and the S card processor is unlatched by inserting a NOP into latch word FPR1. If there are no S cards to be processed, the latch word to the D and E card processors are also set to NOP at this time. The label word is completed by inserting the value of GTNO into bits 2 through 6.

The input list on the card is processed by calling subroutine SAS3 for each entry. The flow chart for SAS3, which contains SAS9 is shown in Figure 4.19. Upon entry into SAS3, the card column is scanned to determine the nature of the input. If a P is encountered the subsequent numerals are binarized and loaded into TEMP. MATL is loaded into the accumulator and SAS9 is called. This results in the entry of a 1 into a location a number of bits equal to the digits following the P, away from the beginning of the interconnection words. For example, if the input list entry is P4, a 1 is entered into the fourth bit of the first interconnection word.

If the input symbol is a G, the binarized digits following the G are augmented by the value of PCTR (the number of P inputs) and entered into TEMP prior to the jump to SAS9. This is done to

move the entry past those positions in the matrix allotted to P entries. As an example, if there are seven P inputs and the input list entry is G13, then a 1 is entered into the fourth bit of the second matrix interconnection word, which is 20 bits (13 + 7) into the matrix line.

If the input list entry is an S, it is important to determine whether the input comes from the set or reset side. In this case, the variable exit word SBIT for subroutine HTBI is set to 1. From the flow chart for HTBI, (Figure 4.16), it is seen that the digits following the S are binarized and doubled. If terminated in an R a 1 is subtracted from the result, if terminated in an S the binarized number remains unchanged. It is this result which, after the addition of PCTR and GCTR, is entered into TEMP, prior to the jump to SAS9. In this case if PCTR = 7 and GCTR = 21, and an entry S8R, is processed, a 1 is inserted 43 bits (7 + 21 + 16-1) into the matrix line (or into the eleventh bit of the third interconnection word.). If the entry had been S8S, the entry would have been a 1 into the $44^{\underline{th}}$ list of the matrix line.

It is important to note that prior to the jump back from HTBI, SBIT is reset to zero. There is no possibility of using the R and S suffix exit unless the main program sets SBIT to 1 prior to

the jump to HTBI.

If an NC entry in the input list is detected, the NOP previously placed in NCX2 insures that SAS9 is bypassed and no entry is made in the interconnection matrix. The B symbol exit shown in SAS3 is used in conjunction with the D card processor and is discussed in that section.

Regardless of the nature of the input, the input counter is incremented by 1 before the exit from SAS3.

After each entry on the input list of the G card is processed, the input counter (INCT) is checked. When it reaches zero, processing is presumed complete and an exit to ESYM is made. Note that if the number of entries in the input list does not agree with the number stated in the gate type an error will be detected. If there are more entries, ESYM will detect an illegal symbol. If there are fewer, the illegal symbol (blank) will be detected by SAS3.

4.7 The S Card Processor

In the discussion of the initial sort it was noted that the detection of an F in the first card column was followed by a test of

latch word FPR1. This word normally causes a jump to ERR1, and is only replaced by a NOP after all the G cards have been processed. Presuming that the G cards are all processed, the third column of the card is sensed. If it is a J, the S card processor is entered at PRCJ; if it is an R, entry is made at PRCR.

The flow chart for the S card processor is shown in Figure 4.21. The separate entry points are used to establish separate initial conditions. In the case of an RS flip-flop only three inputs are available; hence, the input counter (INCT) is set to -3. A 1 is entered into label bit 15 to signify that it is an RS element, and a column is skipped in order to get by the S in column four of the card (for reasons to be soon clarified). In the case of a JK flip-flop INCT is set to -5, and the label is initially cleared.

After these operations, processing is common for both types of flip-flops. The S card counter SNO is incremented and a JMP SKIP is set into variable exit word NCX2. After this, successive card columns are scanned until an S is sensed, or if no S is sensed, until all 80 card columns are exhausted. This S will normally be the first symbol in the S number. Note the difficulty which would have existed had not the S in column 4 of the

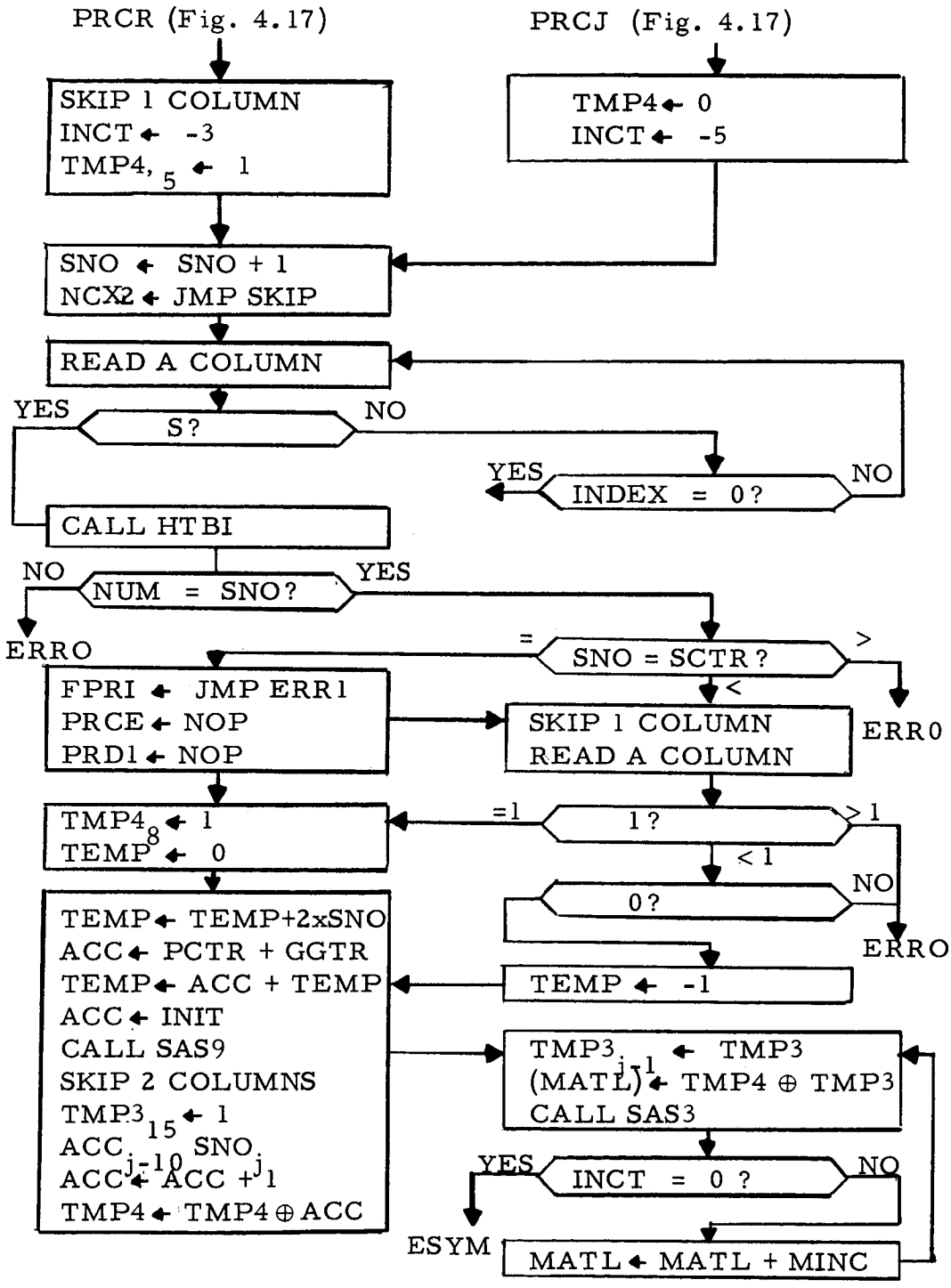


Figure 4.21 S Card Processor

RS cards been skipped. After the S has been sensed, subsequent columns containing numerals are binarized by HTBI. The result, the S number, is compared with SNO to check sequencing. A card out of sequence results in a call to ERRO.

If the card is in sequence, the S number is compared to the expected number of S elements (SCTR). If they are equal, the S card processor is latched by inserting a JMP ERRI into FPR1, and the D and E card processors are unlatched by inserting NOP's into PRCD and PRCE respectively. If the S number is less than the value in SCTR no changes are made in the latching arrangements.

The next part consists of processing the initial status. A column, containing a "(" , is skipped and the next column is sensed. The sensing of a 1 causes label bit 8 to be made a 1, and the initial status is entered into INIT. Entry is made in the manner discussed in the P card processor. It is well to note here, that the S cards have two possible outputs one of which will always be 1 so that the question here is not should a 1 be entered into the proper bit in INIT, but rather into which bit should the 1 be entered. If the initial status is a 1, the entry is made a number bit equal to PCTR + GCTR + 2 x SNO into INIT. If the initial status

were 0, the entry is made one bit to the right.

After the initial status processing is complete two columns are skipped, a ")" and a blank, before the input list is encountered. The input list is processed by SAS3 as it was described in the discussion of the G card with the following modifications:

(a) As each entry in the input list is processed, the matrix line index (MATL) is incremented by MINC. This causes a new line to be created for each input.

(b) Since no matrix line is desired for an unconnected input, the JMP SKIP inserted into exit-word NCX2 causes the MATL to be decremented by MINC before exiting from SAS3.

Each line requires a label which specifies the input composing the matrix line. This is accomplished by inserting a single 1 into bit 15 of dummy word TMP4. Before each entry into SAS3 this word is shifted left once and inserted into the label. It is worth noting, that in the case of an NC input, the previously written label word is erased so that no confusion results.

Processing is complete when the input counter has been incremented to zero. An exit to ESYM is made. Errors are detected if there are not exactly 3 entries in the input list for an RS element, or 5 entries for a JK element.

4.8 The D Card Processor

After the P, G, and S cards have all been read and formed into the matrix, the D cards are read and the information from them used to form a delay table. Each line in the delay table consists of a label word and one or more words of statistical data.

As was mentioned earlier, the delay table label word consists of the last 8 bits of the label word of a matrix line referring to the same type of element or element input. The label word processing is a part of the main program. The statistical data words will vary in format according to the nature of the statistics and for this reason are processed by subroutines. (See Delay Insertion in next chapter.)

The D card flow chart is shown in Figure 4.22. The subroutine shown there is for a uniform density of delays and is included only as an example. Following the detection of a D in the first column of the card the latchword PRD1 is examined. Only if

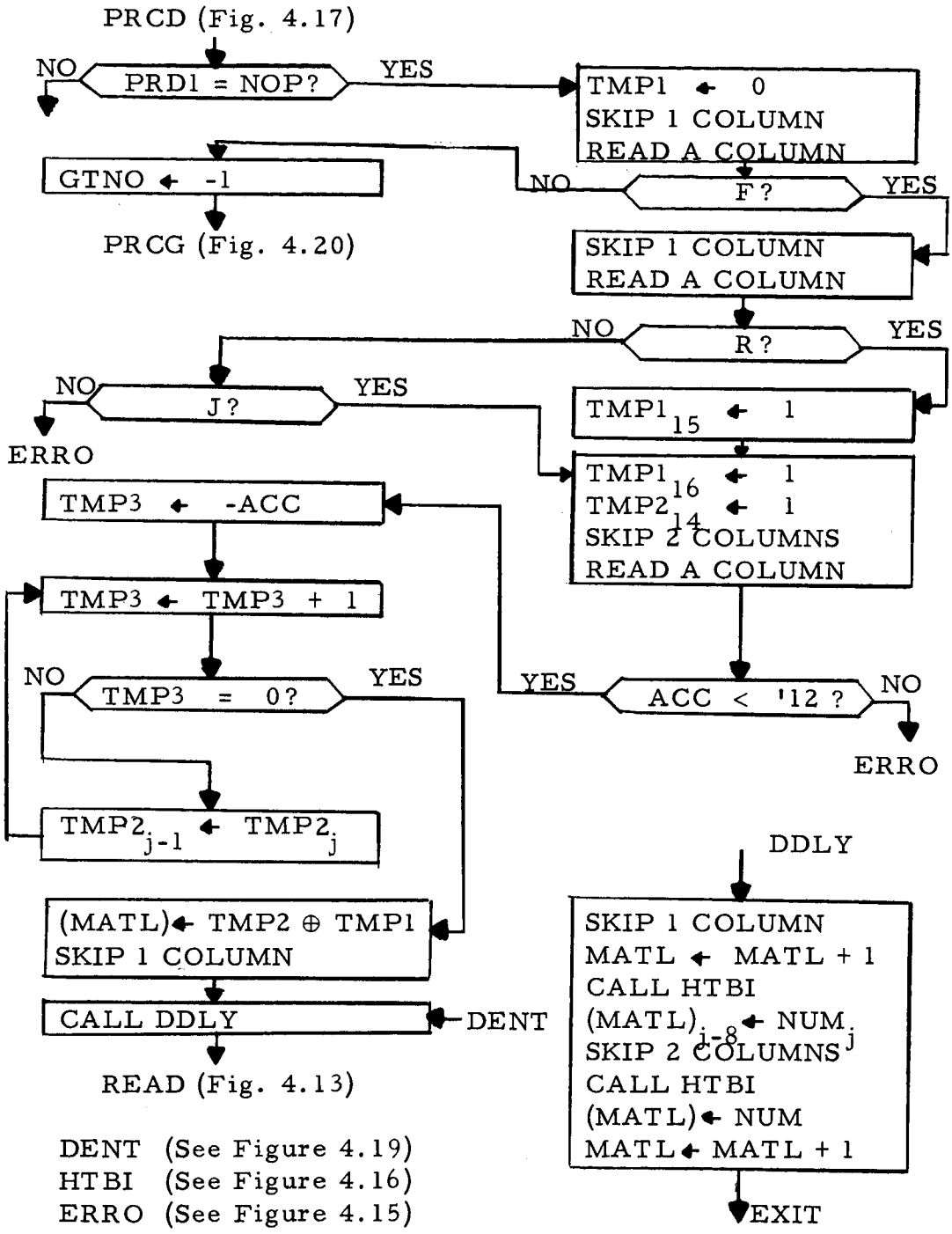


Figure 4.22 D Card Processor

all expected P, G, and S cards are completely processed and no E card has appeared will this word contain a NOP. Presuming that this is the case, the third column of the card is interrogated to see if it contains an F.

Since the label word in the delay table resembles the label word in the matrix, it was hoped that common routines could be used. This proved to be the case for G element delays, but it could not be done conveniently for the storage cards.

If column three contained an F, column five is examined to determine whether it is RS, or JK. If it is an RS element a 1 is placed in label bit 15. In either case a 1 is placed in label bit 16. To obtain the input bit a 1 is placed in bit 14 of dummy word TMP2. After the numeral on the delay card which specifies the input is read, the contents of TMP2 are shifted left a number of times equal to one less than the numeral. The shifted word is then combined into the label.

If the third column of the D card did not contain an F, the value -1 is inserted into GTNO and an exit to the G card processor is made. The value of -1, ensures that the G card latch word will be bypassed and that the GTNO (zero after being

incremented) agrees with the blank following the symbol G on the D card. The card is processed as if it were a G card until the call to SAS3 to process the input list. SAS3 detects a B on the D card where it would expect a P, G, S, or N on the G card. This results in an exit to the D card processor at DENT.

At this point subroutine DDLY is called. This subroutine, contained on a separate tape, creates the statistical data words of the delay table. In the example of uniform density, the card contains the information B = XXX, W = YYY where XXX is the shortest value of delay, and YYY is the width. Upon entering DDLY, the address stored in MATL is incremented by one, and in this address a composite word is stored. The 8 most significant digits are XXX in binary, the other 8 digits are YYY in binary. Subroutine HTBI is used for the binarization in both cases. After the word is created and stored, MATL is again incremented by 1 and the next card read.

The D card processing continues as long as D cards are read, and is terminated by the sensing of an E card. The ordering of the D cards is not significant, although time will be saved if the cards of the most common circuits are inserted first so that they will be near the top of the table.

4.9 E Card Processor

If an E is detected in the first column of the card, latchword PRCE is interrogated. This is normally an instruction to JMP to ERR1, however, if all the P, G, and S cards have been processed in a satisfactory manner, it contains an NOP. Presuming an NOP is in PRCE, the first actions are to latch both the D and E processors by inserting a JMP to ERR1 in each of their latchwords.

The flow chart for the E card processor is contained in Figure 4.23. The function of E card processor is to dimension and initialize certain registers to be used in the simulation, and to compute the initial status of the gate elements.

The length and starting address of the initial status register INIT are already known. It is desired to use this information to construct a register equal in length to store the current status of the system. This register is located immediately succeeding the delay table and is given the mnemonic CURR. Subroutine INS (Figure 4.24) is used for this purpose. The current reading of MATL is defined as CURR, and a jump to INS is made. In INS, the index word CURW is set to CURR, and the index word INI is set to INIT + 2. The contents of the address

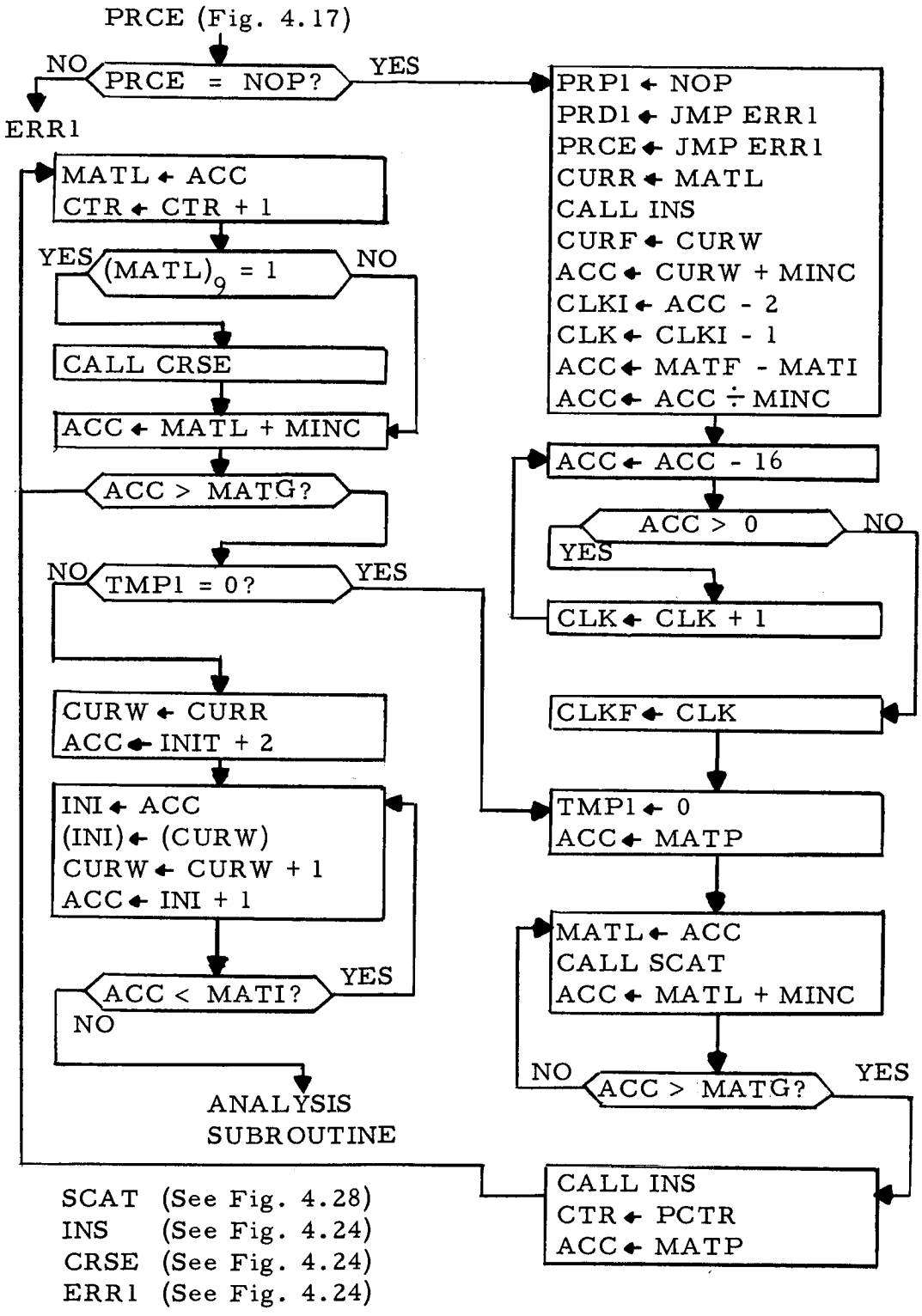
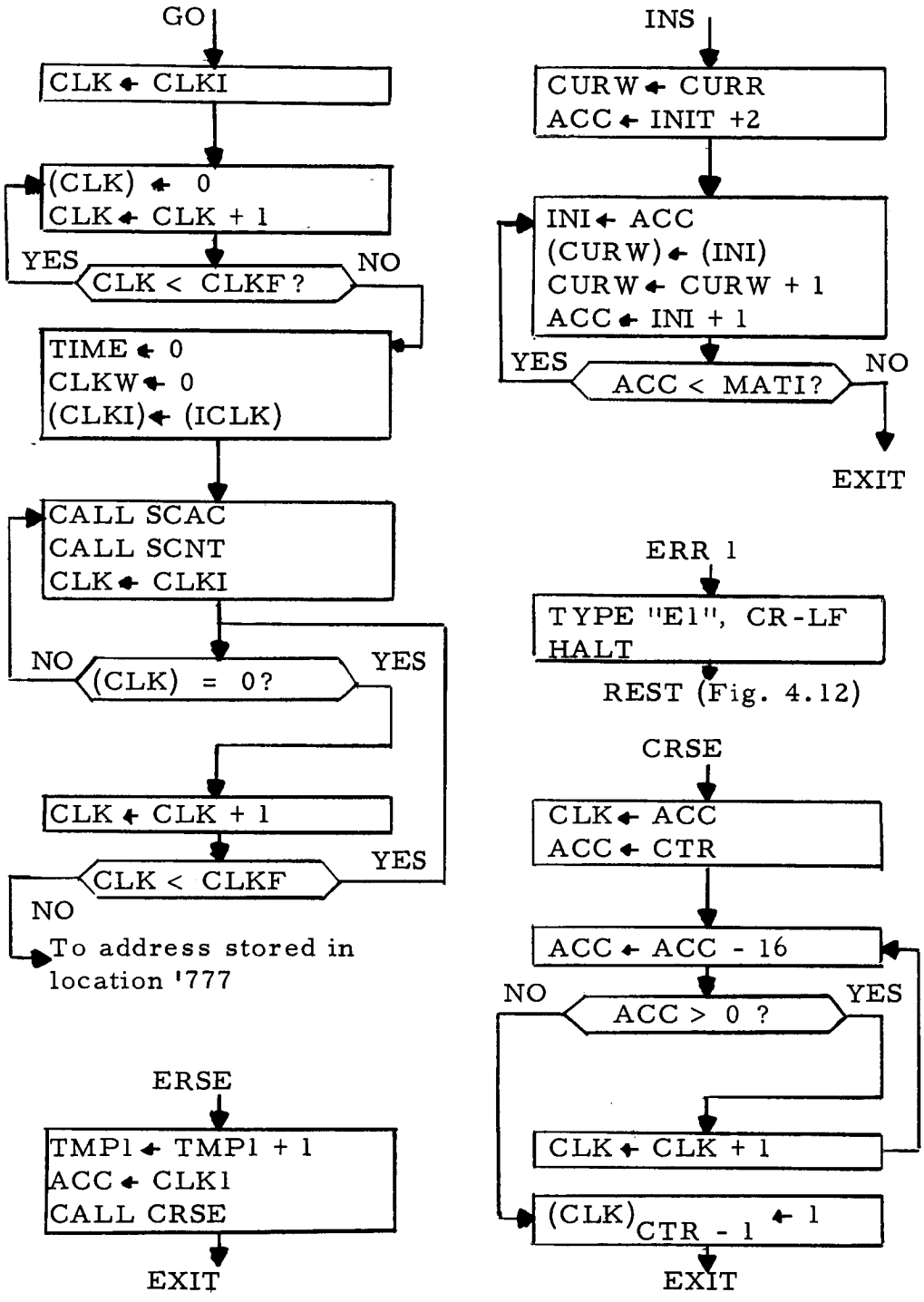


Figure 4.23 E Card Processor



SCAC (See Figure 4.25)
SCNT (See Figure 4.28)

Figure 4.24 Simulator Control Flow Chart and Miscellaneous Subroutines

stored in INI are transferred into the contents of the address stored in CURW. Both INI and CURW are incremented and the process repeated until the incremented value of INI is equal to MATI. The last value of CURW which accepted information from INI is designated CURF.

A second register TRNI is used to store the locations of circuits which are in the transient state. This register is equal in size to the number of interconnection words. It is dimensioned by adding MINC - 2 to the value of CURF. This final word in TRNI is designated CLKI, since it also serves as a marker to indicate the start of the register CLKI.

CLKI is a register which designates which matrix lines contain elements or inputs (In the case of storage elements) which are about to undergo transitions. This register must be dimensioned separately since the number of matrix lines is not a simple function of the number of various types of elements. Its length is obtained by subtracting MATI from MATF, and dividing the result by MINC, yielding the number of matrix lines. This number is then divided up into 16 bit words. The address of that last word used is designated as CLKF. The dimensioning of registers is now complete.

The gates' initial status are then determined. An exit word TMP1 is set to zero, and MATP, (the address of the label word of the first gate word) is inserted into MATL. Each of the G card matrix lines is scanned in turn by subroutine SCAT shown in Figure 4.28. This subroutine examines the label word on each line and causes a transfer to the proper subroutines. In the case of the gates, if the label indicates a NAND or an AND a transfer to subroutine AND is made. In case the label bits indicate an OR or NOR, a transfer to OR is made. These subroutines are discussed in detail in a subsequent section of this chapter. At this time it is sufficient to note that the current status of the inputs are examined, i.e., CURR is compared with the interconnection words in the matrix line. Any changes in the status of the gates required by the inputs in CURR are made. Recall that INIT was previously transferred into CURR and the initial status of the P and S elements can thus influence the gates. Any time a change in gate status is made TMPI is incremented by one.

When all the gate lines have been checked, the changed status is entered into CURR. This entry is made by scanning the label words of all the gates. Any time the status bit is a 1, subroutine CRSE is called. This subroutine generates a variable shift

instruction in the same manner as does SAS9. When the new status of the gates has been entered into CURR the process of scanning the gate lines is repeated. This process continues until a stable situation occurs, that is, no new changes in gate element status are required. This condition is detected by examination of TMP1. This word is set to zero prior to each pass and is incremented only when a change in status occurs. It is examined again after each pass; when it remains zero initialization is complete.

In situations where there is no feedback, the number of passes required to initialize is equal to the number of levels of logic in the path containing the greatest sequence of gate elements.

When initialization of CURR is complete, the contents of CURR are also substituted into INIT so that at this time INIT contains the initial status of all elements. At this point the E card processing is complete, and the system is ready for simulation.

4.10 The Simulation Technique

Once the matrix has been formed, the initial status determined, and the delays have been inserted, the system is ready for simulation. As was noted earlier the simulation is done on a transient basis. This "transient" operation requires a two phased

operation. One phase, a routine with a mnemonic SCAC, consists of the decrementing of the clock words in any matrix line which has a predictable transient in the future. This predictable transient could be a circuit undergoing a propagation delay or an input signal with a transient still to come. After the clock word has been decremented, it is inspected to see if the delay has gone to zero. If so, the output of that element now undergoes a transient and the system status is modified accordingly.

The second phase, with mnemonic SCNT, consists of an analysis of the transients detected by SCAC. If any of these transients cause a change in the inputs of any element, the output is modified accordingly, which may involve the initiation of a propagation delay. If such is the case, an addition is made to list of matrix lines which must be processed by SCAC.

The control of the two phases is shown diagramatically in Figure 4.24. The starting point of the simulation is given the mnemonic GO. Prior to the entry to GO it is assumed that the initial status of the elements has been transferred into CURR. This is done by the E card processor the first time around, and by the reinitializer (Figure 4.35) for all subsequent simulations.

At GO, register CLK is set to its initial value. This register is of variable length. It contains a number of computer words sufficient to allot one bit to each matrix line. The bits from left to right represent the matrix lines in sequence; thus, the first block of bits represents P lines, the next block G lines, and finally the remaining bits represent S lines. CLK has as a function the maintenance of a running list of those matrix lines with active clock words. This includes all signals which still are capable of producing transients and all elements undergoing propagation delays. As an example, if P3 represents a single pulse, there is a 1 in bit 3 of CLK up until the trailing edge of the pulse occurs.

Since the system is assumed initially quiescent, CLK is initialized by loading in the contents of ICLK. This register ICLK, (It will be recalled from the P card processor) contained a list of all input signals except fixed levels and hence is equivalent to the initial status of CLK.

After CLK has been initialized, the elapsed time counter TIME is set to zero along with a clock increment word CLKW. There then follows a transfer to SCAC, followed by a transfer to SCNT. This alternation continues until halted by

either of two conditions:

(a) the desired measurement is made, or

(b) no further transients are expected as evidenced by an all zero content of CLK.

In the first case, the system is reinitialized and, if needed, more simulations are initiated with new delays. The second case results in a transfer to whatever address is stored in location '777. This provides an option for the user to incorporate in his analysis program.

4.11 The Clock Scan Routine (SCAC)

The flow chart for SCAC, shown in Figures 4.25, 4.26, and 4.27, indicates that the first operation is an inspection of CLK to determine which bits contain 1's, thus determining those matrix lines containing active clock words. In the regard, word CTR is used to store one less than the binary equivalent of the bit position of the 1 in CLK.

The first bit position in each word of CLK is treated separately because that bit corresponds to a sign bit in the DDP-116 word and is not subject to the normalize instruction. The remain-

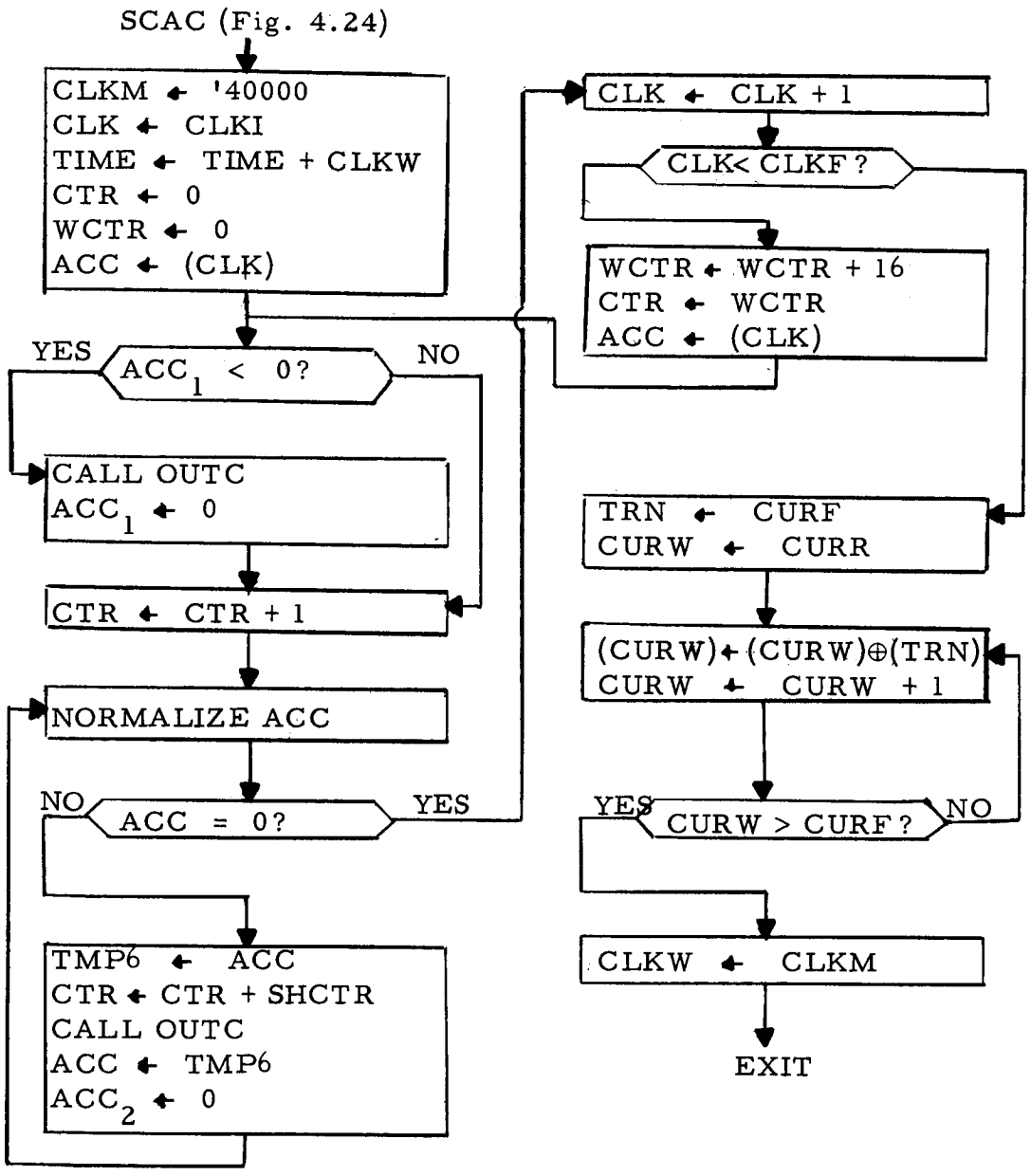


Figure 4.25 Subroutine SCAC

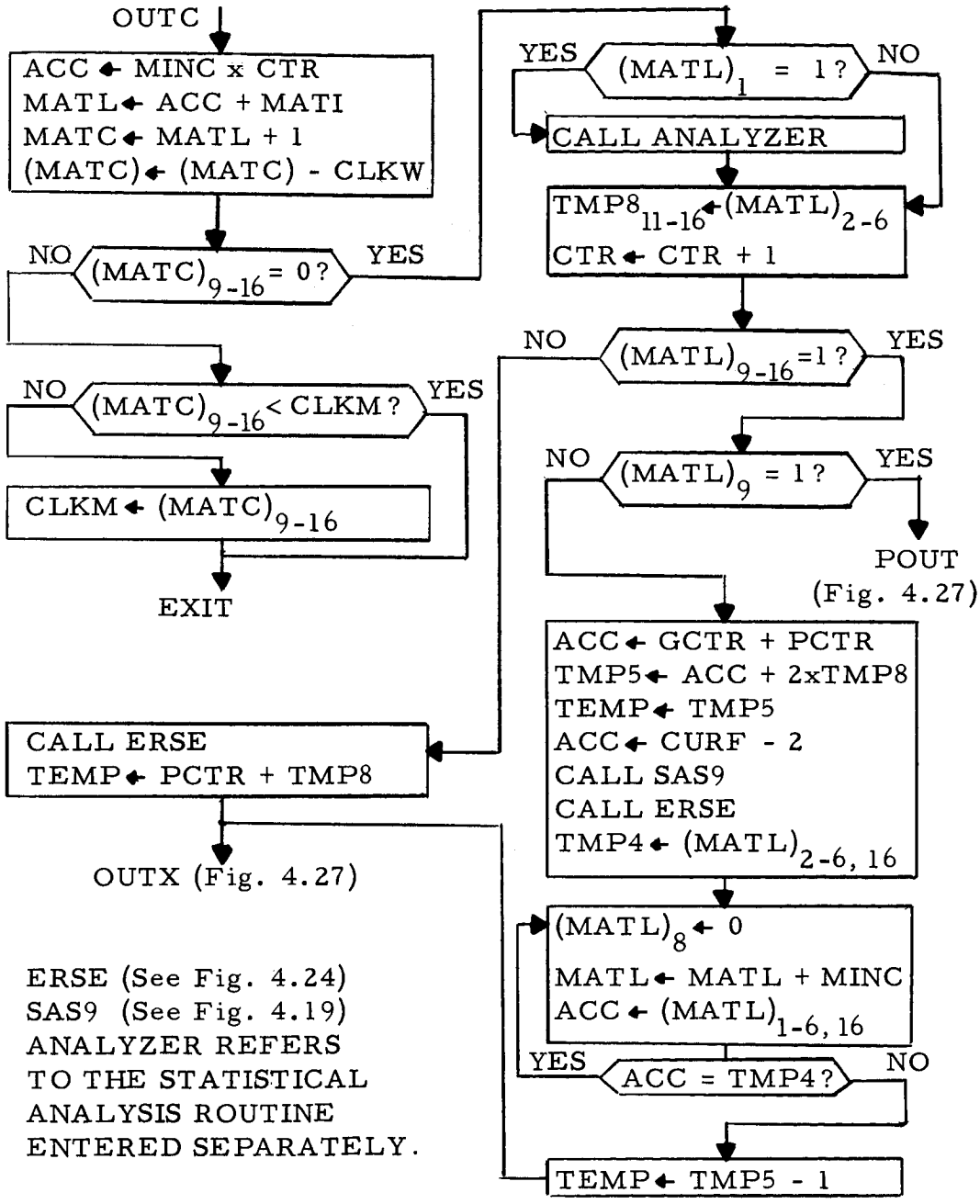


Figure 4.26 Subroutine OUTC (Part I)

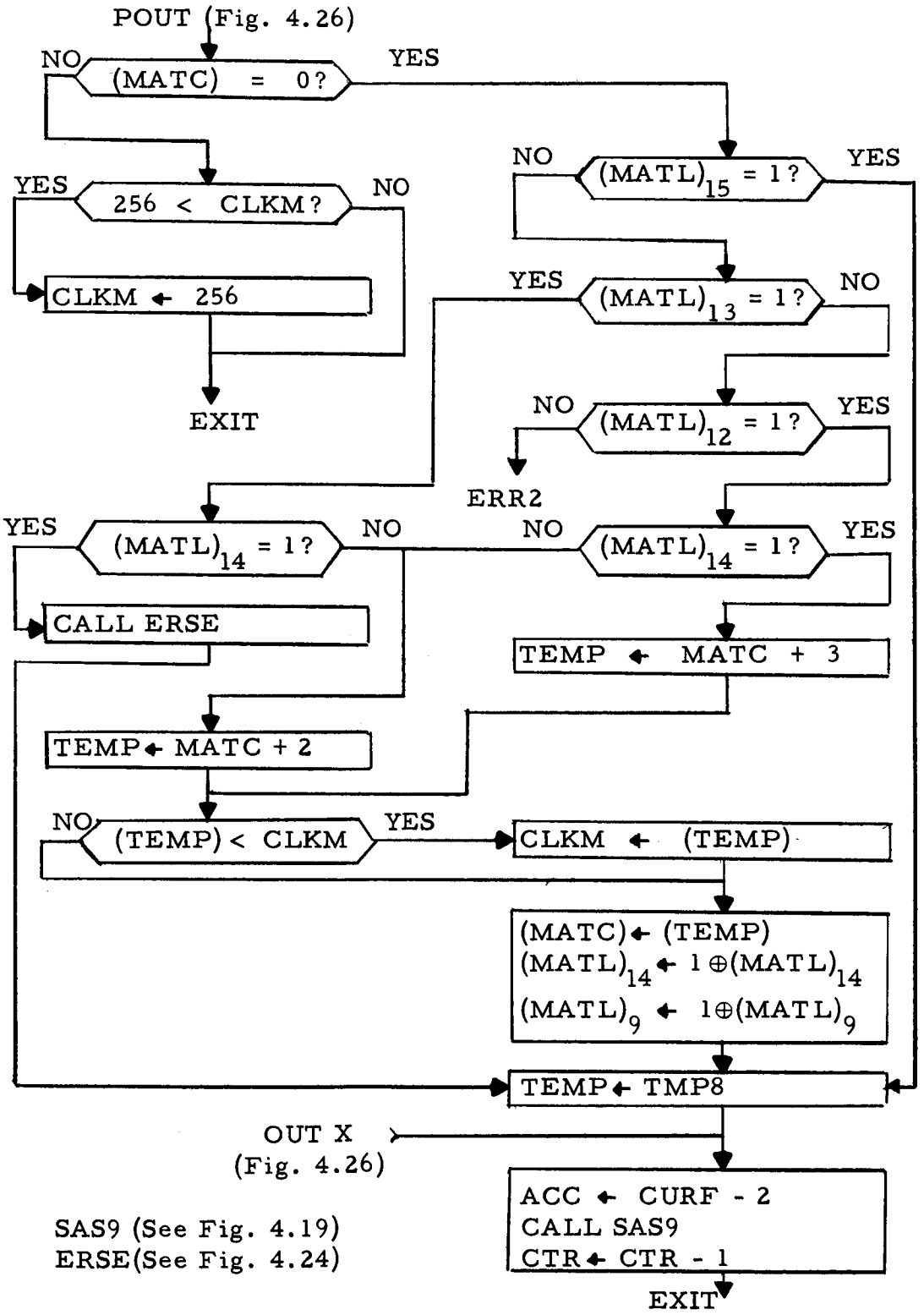


Figure 4.27 Subroutine OUTC (Part II)

ing bits are examined by the normalize instruction which causes a left shift of the accumulator until a 1 is sensed in bit position two. The number of shifts required is held in SHCTR. Note that if the initial 1 in a word were in bit position k, only k-1 shifts would be required. For this reason, CTR is increased by 1 the first time the normalize instruction is used in processing a particular word.

Each time a 1 is detected in a bit of CLK, an exit is made to OUTC (See Figure 4.26). In OUTC, the first step is the subtraction of a time increment (CLKW) from the clock word of the matrix line under examination. The first 8 bits are masked out and the result is compared with zero. If it is not zero, the result is compared with a word CLKM. Any time the subtraction result is less than the contents of CLKM, it replaces those contents. Thus, at the end of a pass through SCAC, CLKM contains the next time increment and is accordingly entered into CLKW. To ensure this operation CLKM is set to 32, 273 prior to each pass through SCAC. On the first pass of the simulation CLKW is set to zero so that the only result of the first pass is a determination of the next time increment.

In those cases where the subtraction of a time increment results in a zero in the last 8 bits of the clock word, the subsequent processing varies, depending on the nature of the line under examination. The label word is then examined in order to determine whether it corresponds to a signal, storage element, or gate. In every case the flag bit is checked and if the matrix label is flagged the analyzer routine is called.

In the event the line corresponds to a gate, the zero results indicate that a propagation delay has elapsed and the gate output undergoes a transient. This requires the erasure of the 1 in the matrix line position of CLK, and the entry of a 1 into the corresponding element position of a register called TRN. Register TRN is a register with the format of the interconnection words of a matrix line and its function is to maintain a list of circuit outputs undergoing transients to be used in connection with SCNT.

Entry of the 1 into TRN is accomplished by subroutine SAS9. The gate number, stripped from the label word, is added to PCTR and stored in TEMP, while the accumulator is loaded with the address stored in CURF decremented by 2. Recalling the operation of SAS9 (Figure 4.19) this ensures entry into the register addressed by CURF.

The erasure of the 1 in CLK is accomplished by a similar subroutine ERSE. ERSE operates in a similar manner to SAS9 with CTR playing the role of TEMP and CLKI being inserted in the accumulator. The result is, however, the "EXCLUSIVE-ORing" of a 1 into the desired bit position. This results in an entry if the bit were a O, and an erasure if it were a 1.

Should the matrix line in question be that of a storage element the processing is essentially the same, with two additions. First, since storage elements have two outputs, both will undergo a transient, so that a 1 is entered into both bit positions of TRN which correspond to that element. This is done by two calls to SAS9. Additional speed, here, could be obtained by an externally imposed variation on SAS9; if events prove this desireable it will be done. Secondly, for reasons to be explained in connection with the RS and JK subroutines it is necessary to set to zero the transient bit in <u>all</u> the matrix lines corresponding to that storage element.

The processing is somewhat more complicated if the matrix line refers to an input signal. First the fact that the last 8 bits are zero following the subtraction of the time increment does not mean that the entire clock word is zero. Recall that in

the matrix line the entire clock word is available for timing input signals. For this reason, the first step in P line processing at POUT is to check the entire clock word for zero. If it is not, then the signal is not ready to change state. In this event the contents of CLKM are compared with 256 and the smaller number replaced in CLKM. This ensures that if any signals are still active, the next time increment will not cause a P line clock word to go negative.

If, however, the entire clock word were zero, then the signal will undergo a change. The label is again inspected. If the signal represents a level change only, a 1 is entered into the corresponding bit of TRN and the corresponding bit in CLK is erased (No more transients on this input!).

If the signal represents a single pulse the processing depends on whether the leading or trailing edge transient is under consideration. The presence of a leading edge transient is indicated by a zero in bit position 14 of the label. This bit is always initially zero. During the first time this line undergoes a transient it is set to 1 so that the next time it will indicate trailing edge. A leading edge of a pulse results in the contents of storage word two (the pulse duration) being entered into the matrix line

clock word. A 1 is entered into the proper bit of TRN but the 1 in the P-line bit position of CLK is not erased. On the second pass the trailing edge is treated just like a level change which, for practical purposes, is what it has become.

If the signal represents a periodic pulse, bit 14 of the label is used to indicate which of the storage words are to be copied into the clock word. If bit 14 is O then the contents of storage word two (D1) become the contents of the clock word, otherwise, the contents of storage word three (D2) are used. In either case a 1 is entered into the proper bit of TRN; and in no case is the 1 in the corresponding bit of CLK erased.

After all the matrix lines indicated by CLK have been examined, the contents of TRN are "EXCLUSIVE-ORed" with the contents of CURR, which has the effect of up-dating CURR. At this point the processing by SCAC is complete.

If the tests fail to indicate a pulse type, transfer is made to subroutine ERR2. This causes the machine to halt after typing out "E2". In contrast to ERR1, ERR2 can be entered in many ways. For this reason the address from which the entry to ERR2 was made is stored in the A Register of the computer console and can be used

in error analysis. Table 4.1 shows a list of A register readings and the corresponding reason for entering ERR2.

It is well to note in connection with the preceding discussion that each time a new propagation delay is indicated in SCNT, the value of this delay is compared with CLKW. If it is smaller, it will determine CLKW for the next pass through SCAC.

Another point worth discussing is the use of the register CLK. It may be argued, at least for small systems, that it is faster to scan each matrix line clock word in turn, checking each word for activity. This argument has greater validity if many transients occur simultaneously. To date, the experience indicates that simultaneous transients are relatively rare and it is believed that the present method is faster.

4.12 The Transient Scan

The second phase of the simulation, routine SCNT, consists of an examination of the contents of TRN, and, if necessary, a comparison of the contents of TRN with the interconnection words of the matrix. It will be recalled that those matrix lines correponding to external inputs contain no interconnection words; thus, only the G and S lines are involved in SCNT.

"A" Register Reading (OCTAL)	Cause of Error
2430	No Input Bit Designated RS Flip-Flop
3027	No Pulse Type Designated P line
3111	Improper Value of MATL During Reinitialization
3467	No Input Bit Designated JK Flip-Flop
3604	No Input Bit Designated JK Flip-Flop

Table 4.1 "A Register" Reading for ERR2 Outputs

The flow chart for SCNT is shown in Figure 4.28 along with the flow chart for the auxilliary routine SCAT. The initial step is an examination of the first word of TRN. If it contains all zeroes, the next word is examined, and so on until all the words of TRN are checked.

If any word in TRN is not all zero, it is compared with the corresponding interconnection word of all the S and G lines. Word WINC, which is incremented by one, each time a new word from TRN is examined, is used to assure that only the corresponding interconnection word is compared with the word from TRN. If the words being compared have a 1 in common, then the circuit associated with the interconnection word has a transient on one of its inputs. This results in a call to routine SCAT.

Subroutine SCAT has the job of determining tye type of element associated with the matrix line in question and of transferring control to the proper processing routine. The interrogation of the label is accomplished by rotating the label word one bit to the right which enables the nature of the matrix line to be completely specified by the first and last bits of the rotated words. A 1 in the first bit indicates a storage element. The last bit now indicates the combinatorial function or the nature of the storage

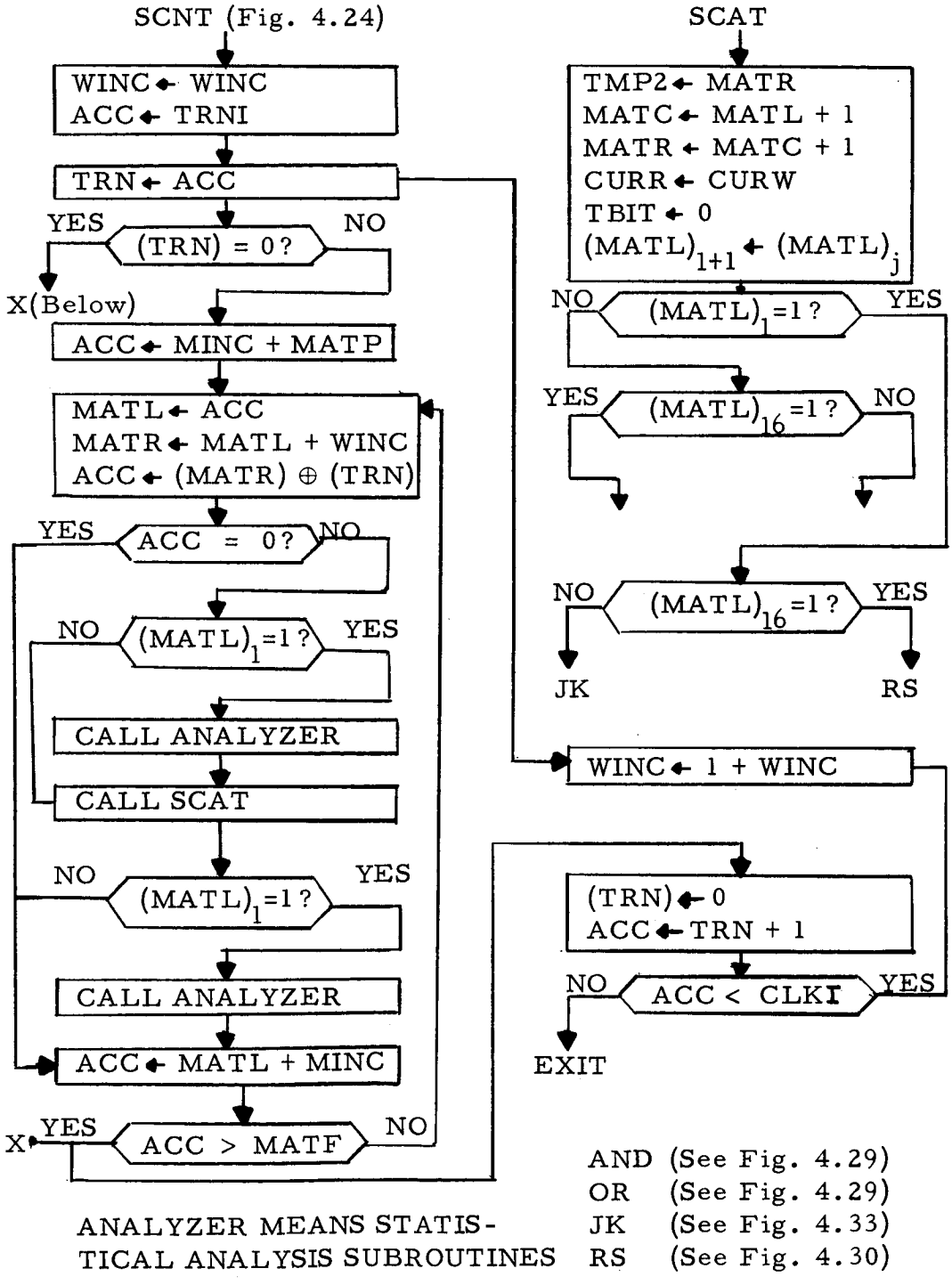


Figure 4.28 Subroutines SCNT and SCAT

element accordingly as the first bit is O or 1.

Once the label has been interrogated control is transferred to the proper subroutine which processes the matrix line in a manner to be discussed in detail subsequently. Any system element which has its state changed results in a modification of CLK.

When each of the words of TRN are completely processed, it is set to zero if it is not already zero. Thus prior to the return jump to SCAC, TRN is all zeroes and CLK reflects the current status of those elements which will undergo changes.

Prior to and subsequent to the call to SCAT the label is interrogated for flags. Any flags result in a call to the statistical analyzer.

4.13 The And-Or Subroutines

The AND-OR subroutines are designed to examine the inputs of any combinatorial element and modify its state accordingly. They are used in two phases of the program. In initialization (Figure 4.23) each gate in turn is scanned by the AND-OR subroutines and the output modified as needed. In normal

operation the gates are scanned only if the transient register (TRN) contains a 1 in a position also contained in the input matrix line of that gate. Thus, normally the gates are only checked by the AND-OR routines if one or more inputs undergoes a transient.

The AND routines will be discussed first since they contain routines used in common with the OR routines.

Any AND or NAND can be in one of the following four states at any time:

(a) the ON state. The ON bit in the label word is 1 and the status bit is 1 or 0 depending on whether the gate is AND or NAND respectively. This corresponds to the case where all inputs are at 1 and any propagation delays have elapsed.

(b) the ON-GOING state. The label is the same as that of the ON state, however, the least significant 8 bits of the clock word are not zero. This is the case where a circuit has just had all inputs set to 1 and the propagation time has not elapsed.

(c) the OFF state. Here the ON bit is zero and the status bit of the label word is 0 or 1 depending on

whether the gate is AND or NAND respectively. The last 8 bits in the clock word are zero. In this state, at least one input is not a 1.

(d) the OFF-GOING state. The label is the same as the OFF state but the last 8 bits of the clock word are not zero. This corresponds to a transition from ON state to OFF state during the propagation delay.

It is necessary to postulate a set of "ground rules" for the performance of AND gates. These rules are somewhat arbitrary and it may be possible that certain sets of circumstance will require modifications of these rules:

I. If an input transient makes all inputs 1's, and the circuit is in the OFF state, it is changed to the ON-GOING state by the copying of the 8 most significant digits of the clock word (the delay) into the least significant 8 digits.

II. If an input transient makes all the inputs 1's, and the circuit is in the OFF-GOING state, it is changed to the ON state.

III. If an input transient makes all the inputs 1, and the circuit is in the ON or ON-GOING state, no change is made. This rule is included to cover those cases arising in initialization and multiple transients.

IV. If the input transient makes at least one input 0 and the circuit is in the OFF or OFF-GOING state no change is made.

V. If the input transient makes at least one input 0 and the circuit is in the ON-GOING state, it is returned to the OFF state.

VI. If the input transient makes at least one input 0 and the circuit is in the ON state, an OFF-GOING state is initiated by entering the delay into the 8 least significant digits of the clock word.

With these ground rules, the initialization can occur by simulating the presence of a transient and checking each gate in turn.

It is also worth noting that in the ON-GOING or OFF-GOING states the label indicates the future status.

Figure 4.29 shows the flow chart for the AND routines. Entry is always made from SCAT (Figure 4.28) so that MATL, MATC, and the initial value of MATR are all defined prior to the actual entry. First the clock word is examined to determine if the gate is undergoing a propagation delay; if so, steering word TBIT is set to a convenient non-zero value. Then a word by word comparison of the current status register (CURR) is made with the interconnection words of the matrix line.

If all the inputs to the gate are contained in CURR then the transient is such as to make all inputs 1. Then TBIT and bit 7 of the label word must be interrogated to ascertain whether ground rule I, II, or III applies. If label bit 7 is a 1, ground rule III requires an exit. If the label bit is zero, and TBIT is zero then ground rule I requires the initiation of the ON-GOING state; if TBIT were 1 ground rule II requires a return to the ON state. In either case the label bits 7 and 8 are changed. This is accomplished by "EXCLUSIVE ORing" the label word with a word which contains 1's only in bit positions 7 and 8. The only difference in processing occurs with the clock word. If TBIT is zero, the 8

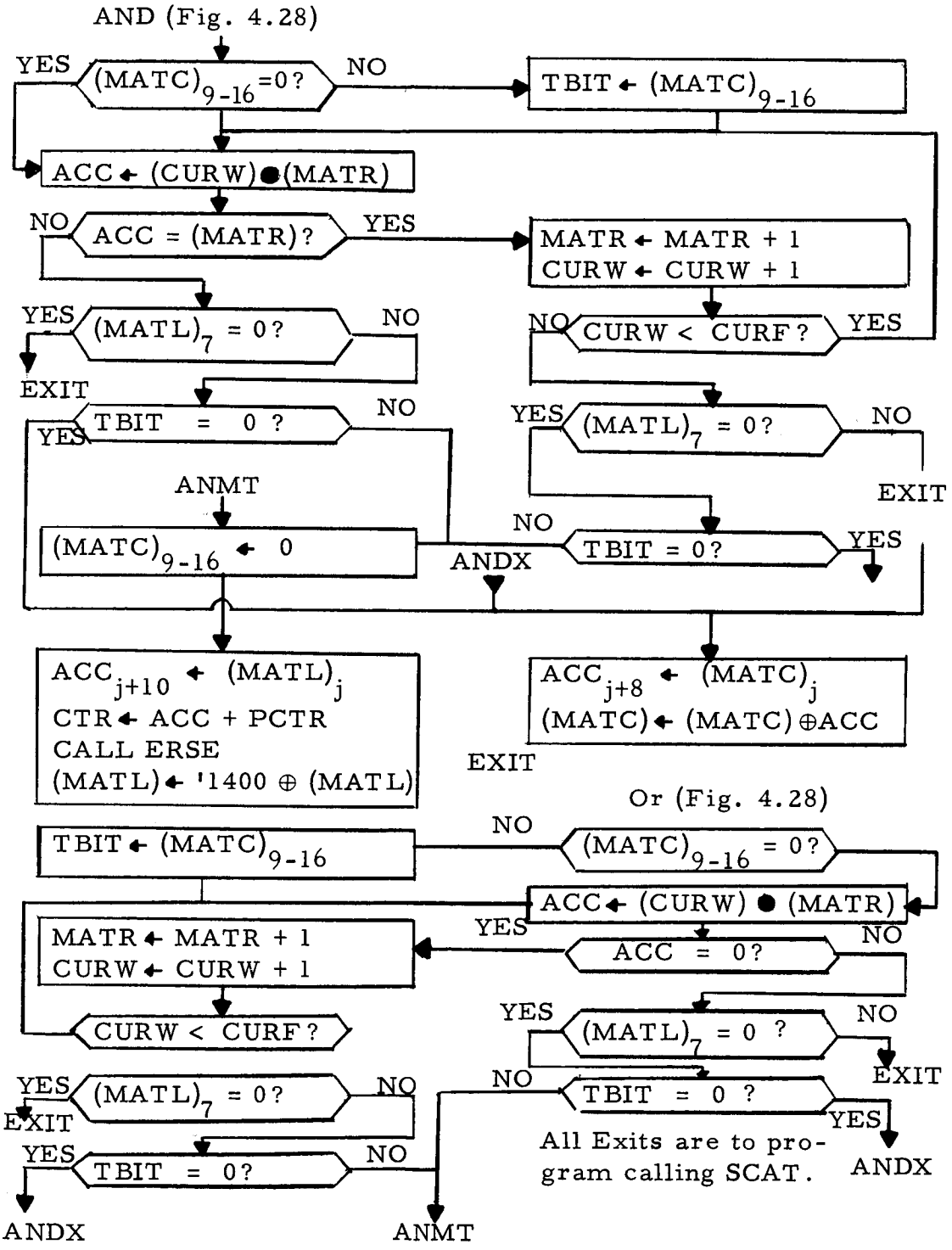


Figure 4.29 The AND-OR Subroutines

most significant bits are copied into the other 8 bits; if TBIT is 1, zero is written into those bits.

In the event that CURR does not contain all the inputs of the gate, then TBIT and bit 7 of the label are examined to determine which of ground rules IV, V, or VI applies. If the label bit is zero, then ground rule IV requires an exit. If the label bit is one and TBIT is 1, then the ground rule V requires a change of label bits 7 and 8 and the insertion of zeroes into the 8 least significant bits of the clock word. The processing under ground rule V is identical to that of ground rule II. Similarly if label bit 7 is 1 and TBIT is zero, ground rule VI can be processed identically to ground rule I.

Each time the state of circuit is changed, a call to ERSE is made. Subroutine ERSE (Figure 4.24) examines the contents of CTR, and EXCLUSIVE-OR's a 1 into the corresponding bit location of CLK. Thus, if an ON-GOING or OFF-GOING state is initiated, a 1 is written into that location; if an ON or OFF state is initiated the previous 1 in that position is erased. Thus the contents of CLK still indicate which matrix lines contain non-zero bits in the least significant half of the clock word.

The OR subroutines operate in a similar manner except that now the states correspond to new input configurations, and a new set of ground rules applies. In the description of the states, the labels and clock bits are exactly the same as for the same state of an AND element.

The states of an OR or NOR element are as follows:

(a) The ON state corresponds to the case where at least one input is non-zero and the propagation delay has elapsed.

(b) The OFF state corresponds to the case where all inputs are zeroes and the propagation delay has elapsed.

(c) The ON-GOING state corresponds to the transition from OFF to ON state during the propagation delay.

(d) The OFF-GOING state corresponds to the transition from ON to OFF state during the propagation delay.

The ground rules for the OR subroutines are listed below:

I. If the input transient makes at least one input take on a non-zero value, and the circuit is in the ON or ON-GOING state, no change is made.

II. If the input transient makes at least one input non-zero and the circuit is in the OFF state, then a change to ON-GOING state is made.

III. If the input transient makes at least one input non-zero, and the circuit is in the OFF-GOING state, then the circuit is returned to the ON state.

IV. If the input transient makes all inputs zero and the circuit is in the ON-GOING state the circuit is returned to the OFF state.

V. If the input transient makes all inputs zero and the circuit is in the ON state, a change to the OFF-GOING state is made.

VI. If the input transient makes all inputs zero and the circuit is in the OFF or OFF-GOING state then no change is made.

Since the initiation of an ON-GOING or OFF-GOING state requires the entry of the 8 most significant bits of the matrix line clock word into the least significant 8 bits, and the initiation of the OFF or ON state requires the setting of zeroes into those 8 bits, and since both cases require a change in bits 7, 8 or the label, the processing can be done by parts of the AND subroutines.

The function of the OR subroutines is then merely to determine which ground rule applies and to transfer to the proper entry point in the AND routines. Thus, the OR subroutines in Figure 4.29 are relatively simple. The last 8 bits of MATC are examined in order to determine if the circuit is presently undergoing a propagation delay. If so, TBIT is made non-zero. Once again CURW and MATR are compared word by word. If any of MATR is contained in CURW, then the transient made at least one input non-zero. If none of MATR is contained in CURW, then the transient made all the inputs zero.

If the transient made the inputs all zero and bit 7 of the label is 0, then ground rule VI requires an immediate exit. If bit 7 were 1 and TBIT were also 1, then ground rule IV requires a change of bits 7 and 8 of the label and the insertion of zeroes in the last 8 bits of MATC. This is done by entering the A subroutines at AMNT. If bit 7 were 1 and TBIT were 0 then ground rule V calls for the insertion of the delay into the last 8 bits of MATC and a change of bits 7 and 8 of the label, actually done by a jump to ANDX.

On the other hand, if the transient made at least one input non-zero, and if the label bit 7 is 1 then an exit is called for.

A combination of label bit 7 equal to 0 and TBIT being 1 calls for a jump to AMNT (ground rule III). The remaining combination of TBIT and label bit 7 both zero calls forth a jump to ANDX as required by ground rule II.

4.14 The RST Storage Elements

The RST elements are treated in a manner similar but not identical to that of the AND and OR subroutines. The major difference is that the inputs to RST element are separate and distinct, and the response to these elements depends not only on the input but on the state of the device.

The states of the RST element may be defined as follows:

(a) The ON state. The set output is a 1. All propagation delays have elapsed.

(b) The OFF state. The set output is a 0. All propagation delays have elapsed.

(c) The ON-GOING state. The device is undergoing a change from OFF state to ON state curing a propagation delay. The status bit is 1 in this case.

(d) The OFF-GOING state. The device is undergoing a change from ON state to OFF state during

a propagation delay. The status bit is 0 in this case.

The ground rules for the RST elements are listed below:

I. Positive transients are necessary, but not sufficient, to activate an input.

II. If the element is in the ON or ON-GOING state any transient on the set input has no effect.

III. If the element is in the OFF or OFF-GOING state any transient on the reset input has no effect.

IV. Any input transient, not eliminated by I, II, or III and arriving when the circuit is in the ON or OFF state, induces a transition to the OFF-GOING or ON-GOING transient respectively.

V. Any input transient, not eliminated by I, II or III and arriving when the circuit is in the OFF-GOING or ON-GOING state, causes a change to the ON or OFF-GOING state respectively.

The flow chart for the RST subroutine is shown in Figure 4.30 and 4.31. The first step is a check on the polarity of

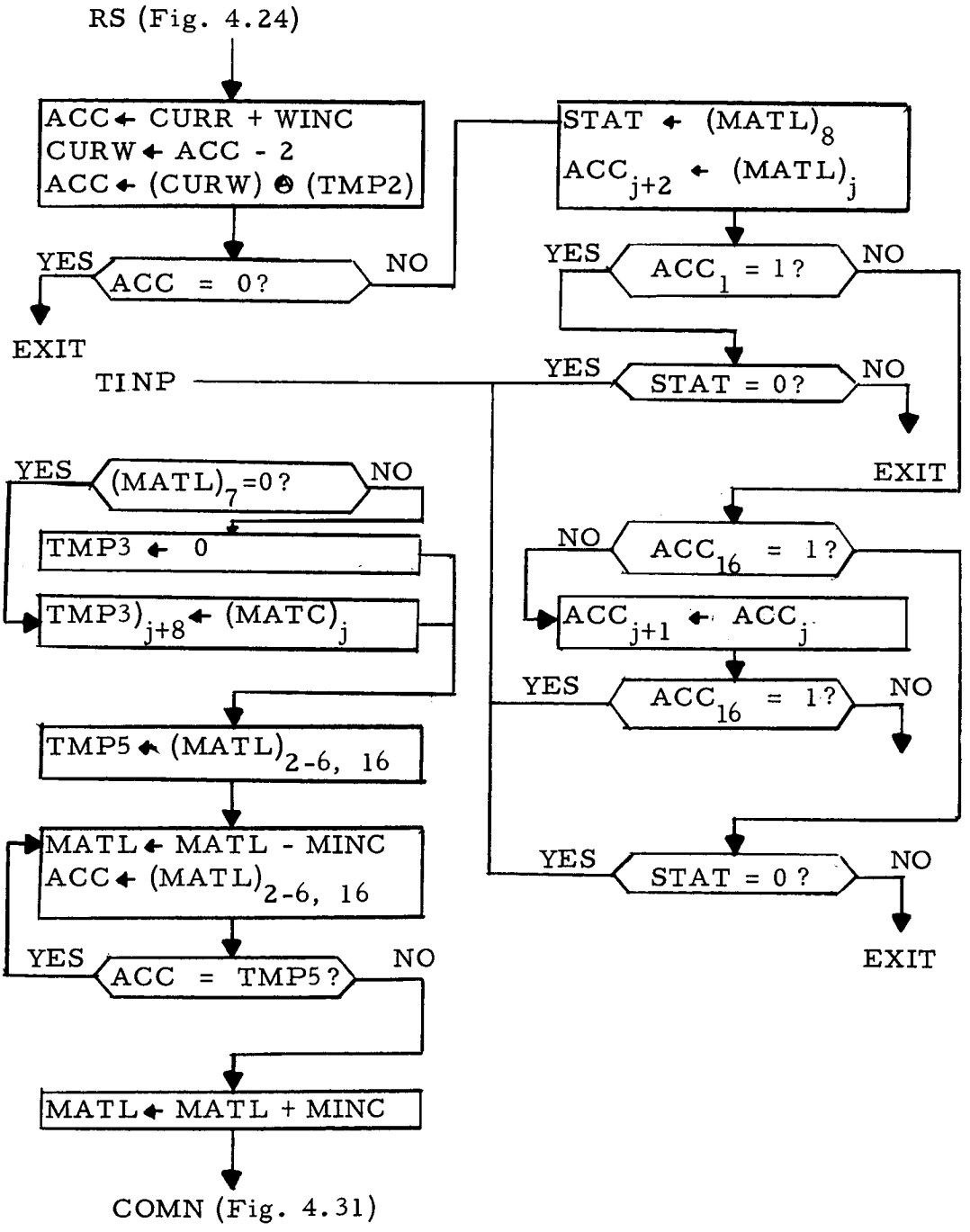
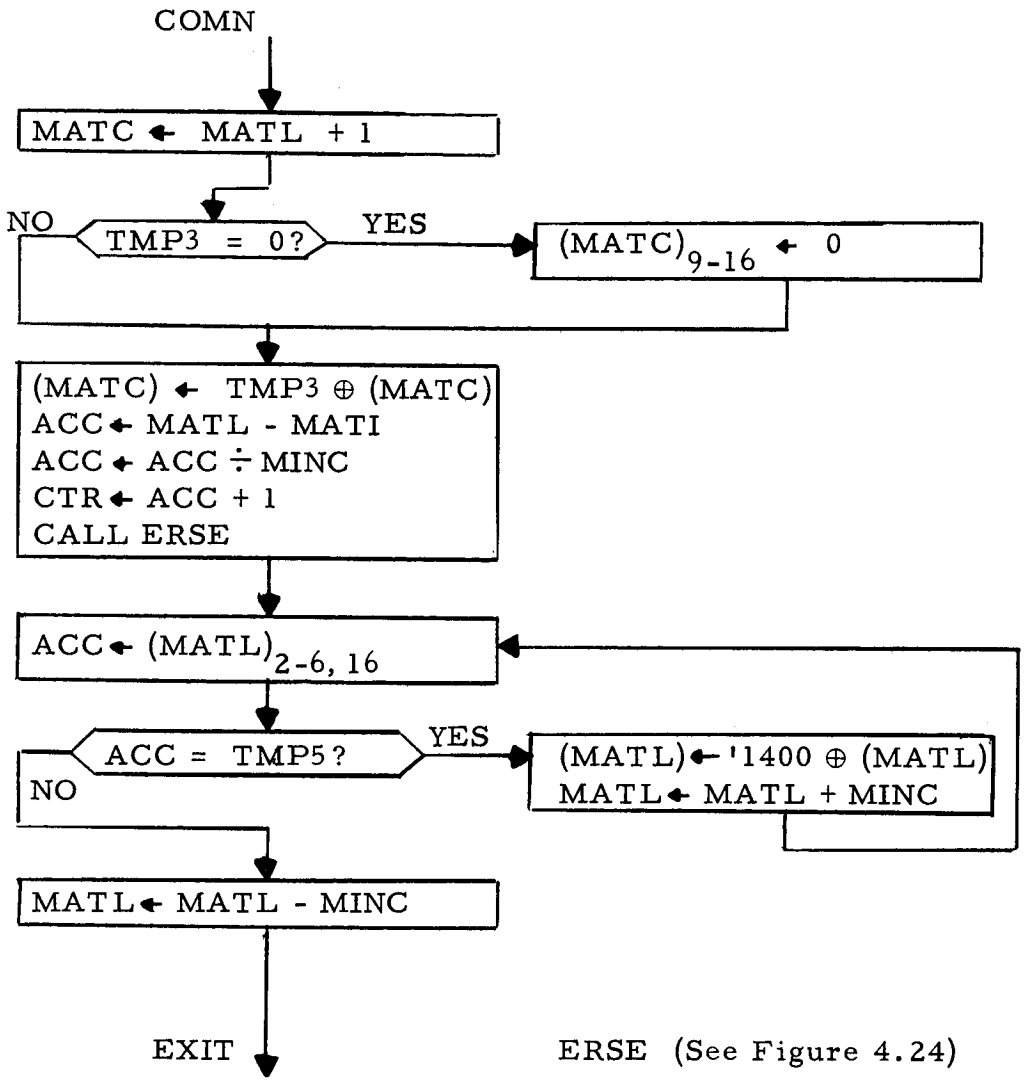


Figure 4.30 RS Subroutines (Part I)



This portion is common to both RS and JK subroutines.

Figure 4.31 RS Subroutines (Part II)

the transient. The proper word of CURR and the contents of the word of TRN containing the transient are ANDed together. The result will be non-zero only if the input were a positive transient. A zero result causes an immediate exit.

The contents of label bit 8 (the status bit) are stored in STAT for use as a steering word. Ground rules II and III are now applied. The label word is rotated 3 units. If the first bit is a 1, the transient is on the reset input. STAT is examined; unless it is a 1, an exit is made. If the last bit of the rotated word is a 1, the set input contains the transient, and unless STAT is zero, an exit is made. A final check is made by shifting right one more unit if neither of the set or reset contained transients. If the last bit is 1 the T input is triggered and processing procedes; if it is not a 1 then the label word is not a valid word and E2 is printed out before a machine halt. (See Table 4.1.)

In the event that the preceding tests indicate processing should continue, the transient bit is examined. If label bit 7 is zero it means that the element is in either ON or OFF state and ground rule IV applies. If this bit is 1, the circuit is either ON-GOING or OFF-GOING and ground rule V applies.

If the transient occurs during ON or OFF state the delay of the input triggered is stored in TMP3; if not, a zero is written into TMP3, and the processing continues.

For convenience in modifying the CLK register it was decided to make only one entry into CLK regardless of the number of matrix lines associated with the RST element. It proves convenient to make this entry in CLK correspond to the first matrix line. To accomplish this, the matrix label words are scanned upward until a new S number is sensed, or a non-S element label occurs.

The clock word of the first matrix line of the S element is then modified following an examination of TMP3. If TMP3 is zero, the last 8 bits of the clock word are set to zero; if not, the contents of TMP3 replace those last 8 bits. The number of the matrix line is computed and stored in CTR. A call to ERSE results in the proper bit of CLK being changed.

The routine makes a down scan now until a word with a different S number is sensed or the all zero "S number" of the first word in the delay table is sensed. At each step of this scan label word bits 7 and 8 are changed. Note that application of

ground rules IV and V requires only that these bits be changed regardless of their prior status. At the completion of the down scan control is returned to SCNT.

A point worth more discussion occurs here. The reader will, no doubt, have noted the relative complexity of the operation which results in a longer processing time. Some of this time would be saved if a fixed, rather than variable, number of matrix lines were used for the RST elements.

This would result in the insertion of several blank lines in the matrix. However, up and down scans could be done without label interrogation; hence, much faster. But an increase in speed would be offset by an increase of matrix size and by an increased time required to check the blank lines during SCNT.

4.15 The JK Flip-Flop

The JK flip flop is treated separately because it illustrates a special point. The JK element to be treated here is not the theoretical JK element described in switching texts, but is the practical JK element manufactured as a microcircuit. This causes a great increase in the problems involved in processing.

The theoretical JK element could be handled in a manner analogous to the RST elements with the inputs in question being only J or K or with the addition of a clock input. The processing would be similar in nature with an inspection of the transient's polarity, and the state of flip-flop yielding the necessary information for any required state changes.

Figure 4.32 shows the logic diagram of a JK flip-flop as presented by Signetics, Inc. This is a monolithic circuit from a single silicon chip. The relatively large number of gates is required since all connections are direct coupled. Recall from the introduction that difficulty in obtaining large capacitances for coupling was a property of the monolithic circuits.

This circuit has five inputs and the output is generally determined only after an examination of all or most of these inputs. The J, K, and C inputs are normally used for synchronous operation, with the truth table shown in Table 4.2(a). The PJ and PK are the preset inputs and operate asynchronously with the truth table shown in Table 4.2(b).

Using the logic diagram of Figure 4.32 switching table of Table 4.3 was obtained. This table gives all the cases where

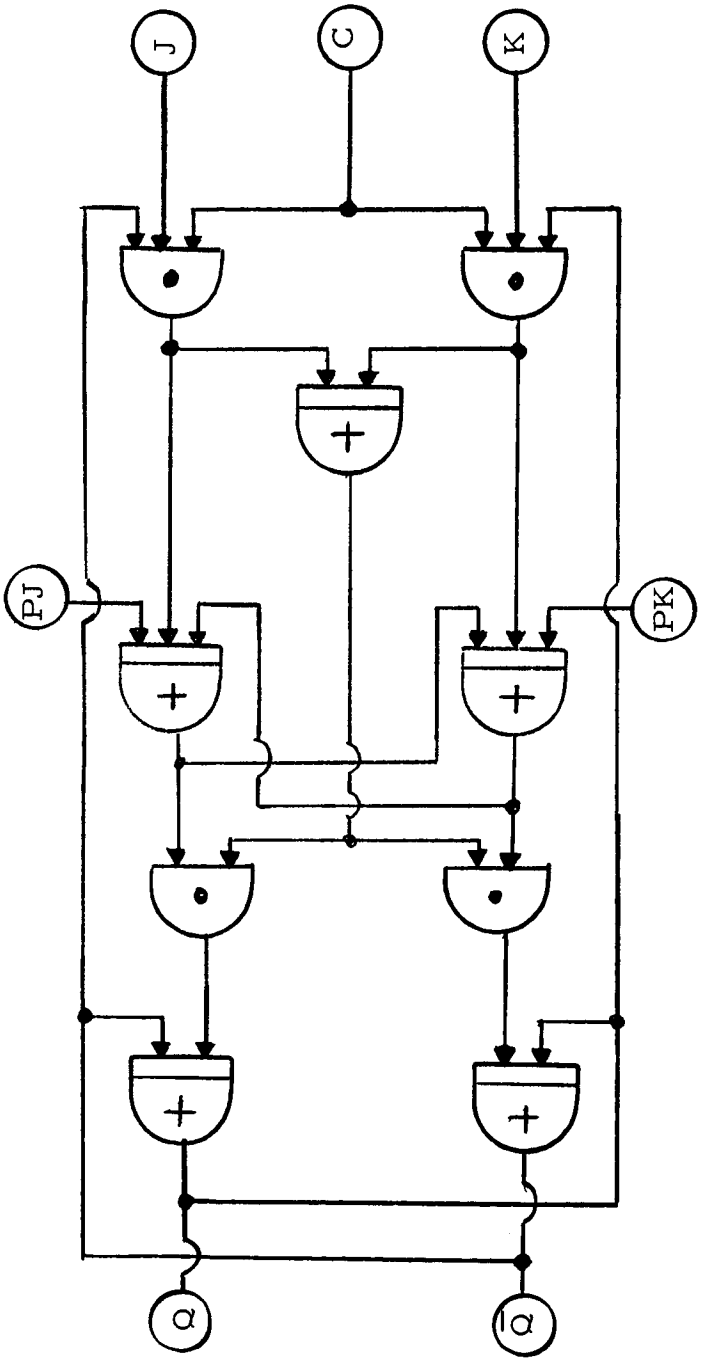


Figure 4.32 Logic Diagram of a JK Element

J	K	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Note: The indicated transitions are initiated by a β transition on C.

(a) Synchronous Truth Table

PJ	PK	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	*
1	1	1	*

Note: When both PJ and PK occur, no change takes place, the final state is determined by which input remains at 1 last.

(b) Preset Truth Table

Table 4.2 JK Element Truth Tables

INPUTS					Initial State	Final State
P_J	P_K	C	J	K	Q_N	Q_N
↑	0	0	ϕ	ϕ	0	1
↑	0	ϕ	0	ϕ	0	1
↓	1	0	ϕ	ϕ	1	0
↓	1	ϕ	ϕ	0	1	0
0	↑	0	ϕ	ϕ	1	0
0	↑	ϕ	ϕ	0	1	0
1	↓	0	ϕ	ϕ	0	1
1	↓	ϕ	0	ϕ	0	1
0	0	1	↓	ϕ	0	1
0	0	1	ϕ	↓	1	0
ϕ	0	↓	1	ϕ	0	1
0	ϕ	↓	ϕ	1	1	0

↓ Transient from 1 to 0 on input.

↑ Transient from 0 to 1 on input.

0 "Zero Level" on input.

1 "One Level" on input.

ϕ Level on input immaterial.

Table 4.3 JK Element Switching Table

an input transient gives rises to a change in state of the device. Much more complicated rules than the case of the RST flip-flop are involved here, and the processing is correspondingly more complex.

The flow chart of the JK processor is shown in Figures 4.33 and 4.34. As in the case of the RST element the first check made is on the polarity of the input transient. From Table 4.3, the only inputs activated by a positive transient are the PJ and PK inputs. If the input transient were positive, then an exit is made if neither PJ nor PK is the input in question. If either is the active input a further test differentiates between them. If the PK input is involved, the status (STAT) of the element is then interrogated, unless it is 1 an exit is made. Similarly if PJ is involved an exit occurs unless STAT is 0. In the event no exits are made, entry into SCBD is made. Note that the input containing the transient is still specified by the contents of TEMP.

In the case of the negative inputs the inputs are then individually inspected. Those inputs requiring special status conditions are checked for those conditions. If they are met

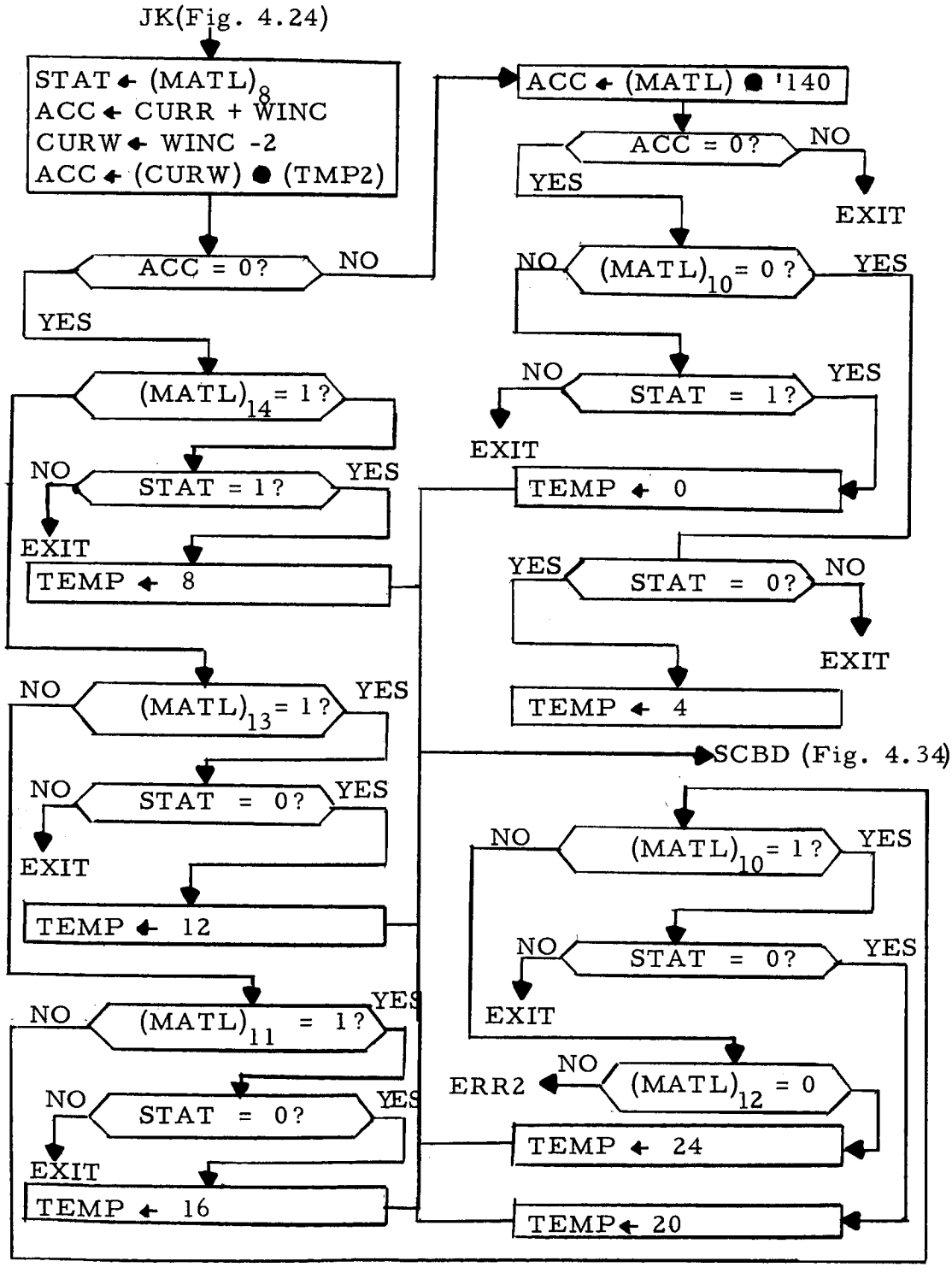


Figure 4.33 JK Subroutines (Part I)

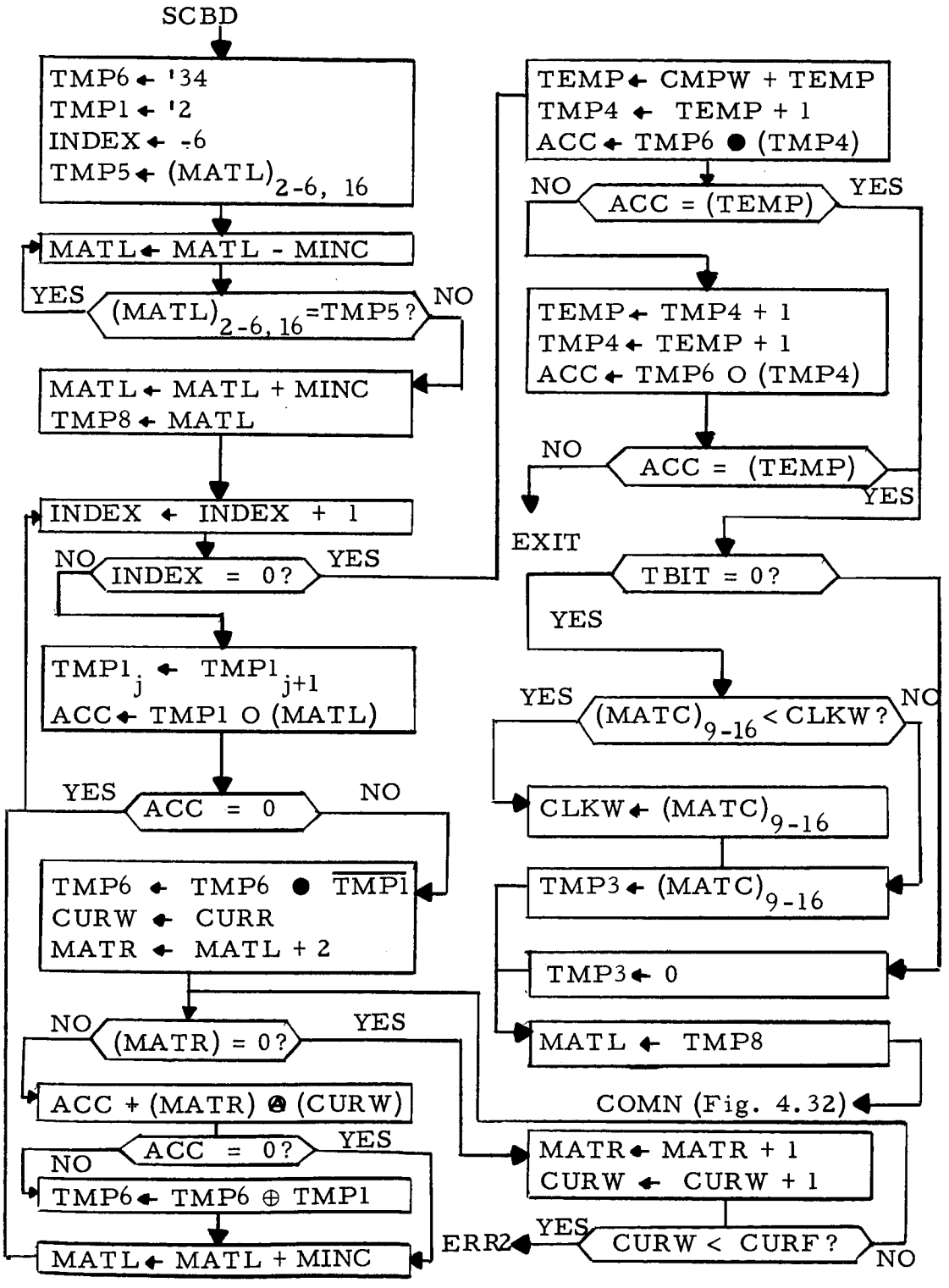


Figure 4.34 JK Subroutine (Part II)

input processing continues in routine SCBD, otherwise an exit to SCAT is made. Again the flow chart indicates that the input containing the transient is indicated by the contents of TEMP.

Figure 4.34 shows the flow chart for routine SCBD. The first step is a scan to locate the first line of the matrix lines common to the JK element. This scan merely indexes back the matrix line and checks the label until a new S number is reached.

After this, during a down scan, a new input word is constructed in TMP6 which gives the status of each of the inputs in the last five bits. This word is initially set to '34 which represents of the effect of non-connected inputs. The first matrix line is examined and its input bit is compared with the input bit for the J input. If there is no agreement, then the J input is unconnected and no change is made in the last bit of the contents of TMP6. If there is agreement, this last bit is made to agree with the status of the J input level. This process continues until all five inputs have been examined.

At this time necessary conditions on the inputs for a change in state of the JK element are compared with the status of the inputs as shown in TMP6. The necessary conditions, which

vary according to which input is excited, are stored in a list called CMPW. The selection of words from CMPW is determined by the contents of TEMP.

There are several "don't care" conditions listed in Table 4.3. Account is taken of these by setting to zero the bits in both TEMP and the comparison word which corresponds to "don't care" inputs. A mask for this purpose is stored in CMPW immediately behind the corresponding comparison word. In most cases, two comparisons are required, so that two comparison words are compared with the contents of TMP6. If only one comparison is needed a dummy word in CMPW is used.

In the event that either comparison is satisfied, TBIT is tested. If TBIT is zero, the delay associated with the input triggered is compared with CLKW and inserted into TMP3. If it is less than CLKW, it also replaces CLKW. If TBIT is not zero, a 0 is inserted into TMP3. The address of the label word of the first matrix line of that element is inserted into MATL, and control is transferred to the RS processor at COMN. All subsequent processing is as described in that section.

The processing involved in the treatment of the JK element is considerably more than in the case of the RST element. As long as the status depends on more than one input the processing time will be much longer. (See Chapter VII for an estimate on one example.) Any time the user can make simplifying assumptions it will usually pay him to do so. For example, if JK elements are connected as binary counter elements, and being used for practical purposes as RST elements, the user should consider calling them RST elements to gain speed. Here the C input could be considered as the T, the PJ input as the R, and the PK input as the S. Note that the RST processor uses positive transitions on T, thus any pulses appearing on C should be inverted, and the counter stages should be driven from the "wrong" side of the driving flip-flop.

4.16 Reinitializer

In repetitive statistical experiments, it is necessary to reset the system to the same set of initial conditions before each experiment. This requirement is independent of the nature of the statistics involved or in the nature of the analysis; thus, the re-initializer is included in the main program. Its calling address with mnemonic REI is stored in location '471 so that it may be accessed by any analyzer. The flow charts are shown in Figure 4.35.

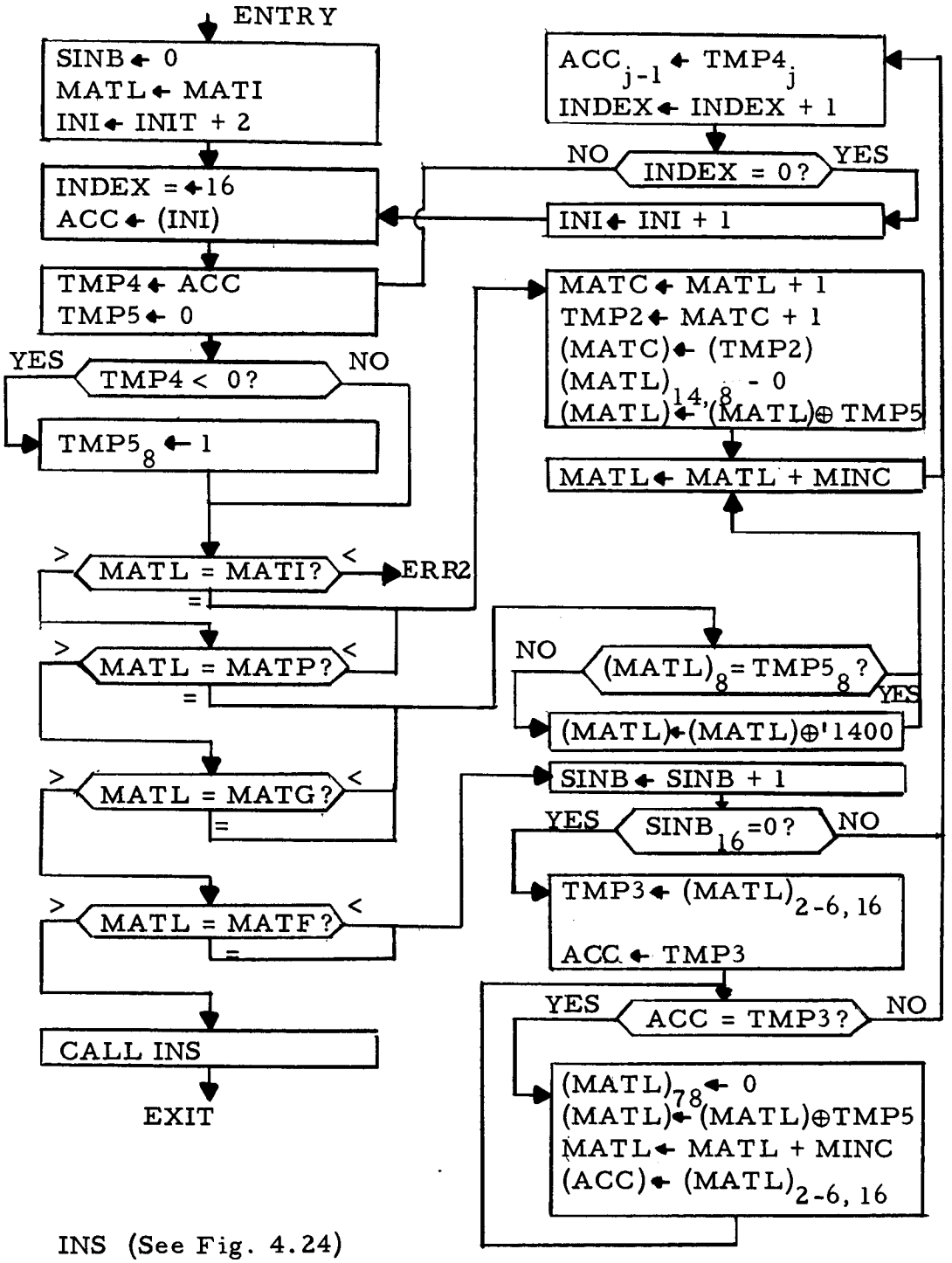


Figure 4.35 Reinitializer Flow Chart

Its operation is based on a bit by bit examination of the register INIT. Initially MATL is set to MATI, a zero is inserted in exit word SINB, and INIT + 2 is set into location INI. The index register is set to -16 and contents of the address stored in INI are set into TMP4. Location TMP5 is initially zero; it is used to hold the initial status bit of the element under consideration. Any time a matrix line label contains 1, TMP5 is also set to 100.

The next step, once the status bit is loaded into TMP5, is to determine whether the line addressed by MATL is P, G, or S line. This examination is done examining MATL, relative to MATI, MATP, MATG, and MATF.

If MATL is at least equal to MATI but less than MATP, it represents a P line. In this case, the contents of storage word one in its matrix line replace the contents of the clockword. The label bits 14 and 8 are set to zero and replaced by the contents of TMP4 respectively. The address in MATL is incremented by MINC and a transfer to NXTL (next line) is made.

Should the value of MATL indicate the line in question is a G line, it is necessary to reinitialize bits 7 and 8, the ON bit and STATUS bit respectively. The status bit of the line label is

compared with the contents of TMP5. If they are the same no change is made and a transfer to NXTL is executed. If they differ bits 7 and 8 are both changed prior to the exit to NXTL.

If the bit in INI refers to an S element, somewhat more complex processing is involved. Recall that INIT holds two bits for each S element and that the status bit in an S line label refers to the set side. For this reason the first bit in INI for a given S element is skipped. This is accomplished by means of an exit-word SINB which was initially zero. Each time an S element bit in INI is examined SINB is incremented by 1. S element reinitialization only proceeds on occasions when SINB is even. In this case a pseudo label for the S element is created by masking out all label bits but the 2nd, 3rd, 4th, 5th, and 16th. All subsequent matrix lines which match those bits have bit 7 (transient bit) set to zero and bit 8 set to the value stored in TMP5. When the label line indicates the next highest S number an exit to NXTL is made.

The next line routine (more properly, next element) causes the contents of TMP to be shifted left once, while augmenting the index register by 1. The index register which here functions as a shift counter can reach zero. If it has not, the preceding analysis is repeated operating on the next bit on INI (first bit of

TMP4). If the index register had reached zero, a new word from INI is inserted in TMP4 and index register is reset to -16.

The exit from REIN is controlled by the value stored in MATL. When it finally reaches the value of MATF the system is reinitialized and an exit is made.

This method of controlling the exit is used as opposed to an examination of INIT, because the last word in INIT will in general be a fractional word, that is, not all its bits will be utilized for status information.

CHAPTER V

THE SUPPLEMENTARY PROGRAMS

5.1 General

It was considered desirable to construct the main program in such a manner as to make it independent of the analysis to be performed and of the nature of the device statistics. In order to accomplish this purpose the analysis is performed by means of flagged exits from the main program, and the delay insertion is accomplished by transfers from the main program.

Several flagged exits are provided in the main program. One is contained in SCAC (Figure 4.25) and is useful in determining the time at which an element output changes. Two other flagged exits are provided in SCNT (Figure 4.28). These are useful in determining the time at which an element state was changed.

The analyses which can be performed using the flagged exits are limited only by the ingenuity of the user. Therefore the descriptions of routines in this chapter are of those which were of most use to the author and those which it was felt would be most exemplary.

5.2 Delay Insertion

After the system has been initialized, and at every time a system is reinitialized for a new simulation, it is necessary to insert delays into the elements. These delays are chosen according to delay statistics of the given element and are inserted into the 8 most significant bits of the clock words in the matrix. Since the method of selecting the delays depends on the chosen statistical model, the delay insertion is done by subroutine which is entered separately. The combined delay insertion routine and that portion of the delay table forming routine which is not in the D card processor are termed the delay package and are loaded together into the machine.

The flow chart for a typical delay package is shown in Figure 5.1. DDLY is that portion which forms the entries in the delay table after the D card has made the line label, and DINS is the portion which inserts the delay. The statistics of the delay in the example shown are assumed to be a uniform density of delays. The operation of DDLY is described under the discussion of the D card processor on page 96 .

Subroutine DINS consists of the examination of each matrix line label in turn. The first 8 bits of the word are masked

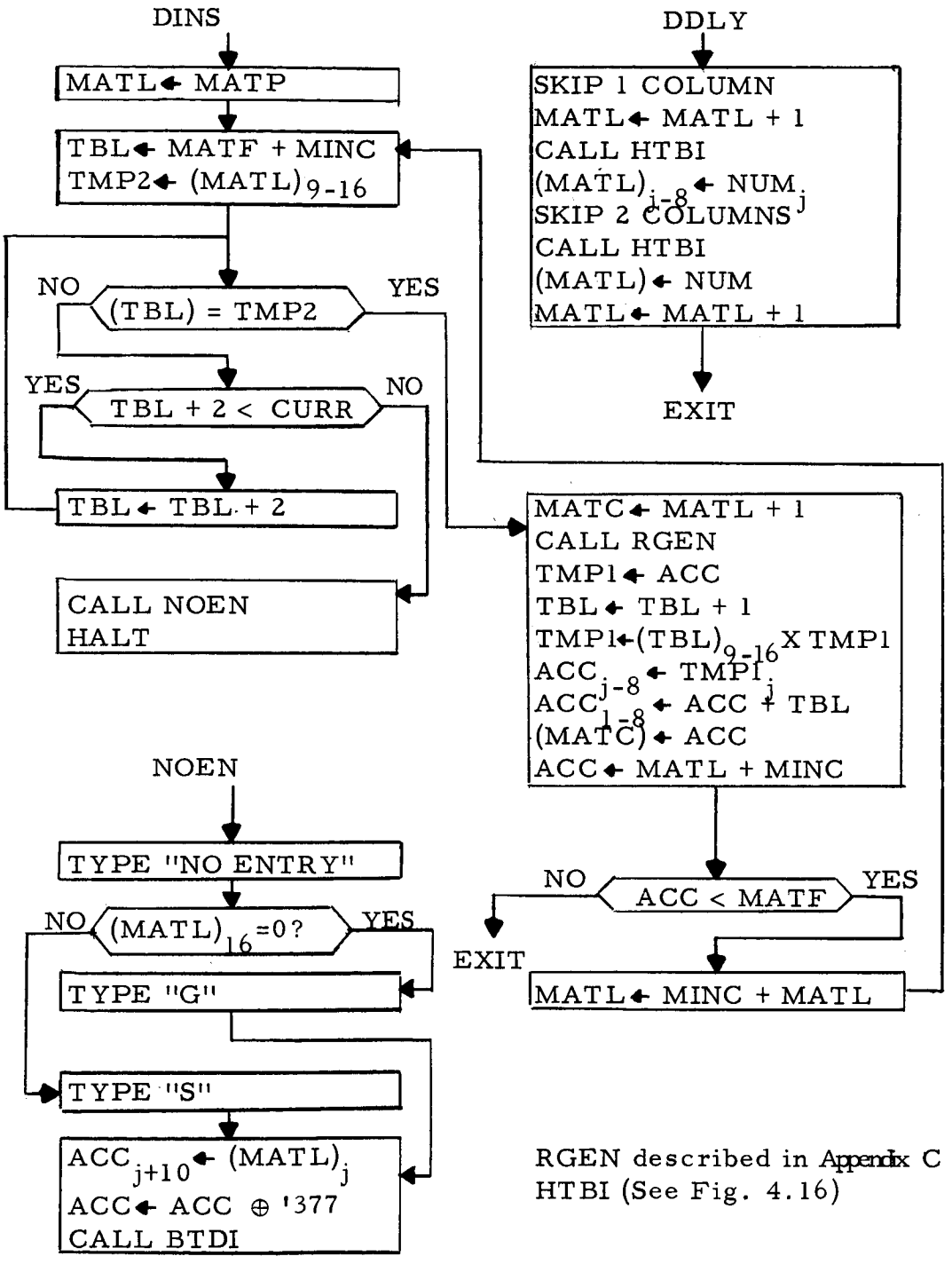


Figure 5.1 The Delay Package

out and the result is compared with each of the delay table label words in sequence. If no comparison is made after scanning the entire delay table, control is transferred to subroutine NOEN which causes G or S number of the element in question to be typed out following the words NO ENTRY.

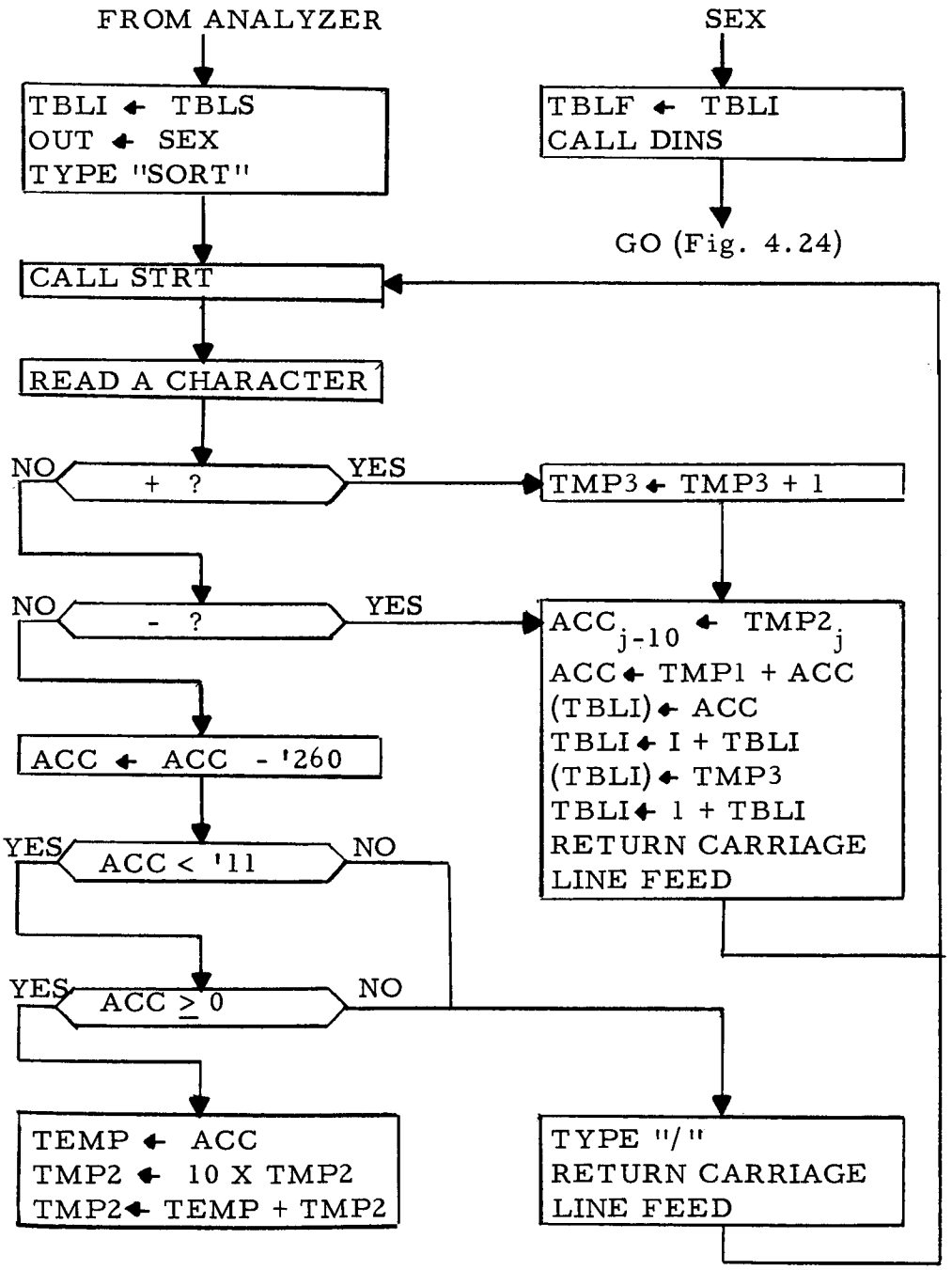
If there is an entry in the delay table corresponding to the element type, a typical delay is created. This delay is governed by the statistics of the element type. For this example it will be assumed that the element type is governed by a uniform density of width W starting at delay B. This continues the example begun in the discussion of the D card processor. It will be recalled in this case, that the delay table entry has the most significant 8 bits representing B and the least significant 8 bits represent W. Delay generation is accomplished by calling the random number generator (See Appendix C.), multiplying the result by the last 8 bits of the delay table entry, rounding the result, and adding it to the first 8 bits. The result is then stored in the first 8 bits of the clock word of the element under examination.

The processing continues until all G and S lines of the matrix have been examined after which a return is made to the calling program.

The NO ENTRY type out will occur on the first time delays are inserted. As the program is presently constituted, a total restart of the matrix maker is required with the addition of the missing D card. Should it be desired, and should more memory space be available, an improvement is possible. Provision could be made for the insertion of only the required D card or cards. This would require a delay table scan before the E card procession, since E card processing closes the delay table and leaves no room for expansion.

5.3 The Sorting Subroutine

The sorting subroutine is a modification of the delay package which permits the operator to select those components to be sorted and includes the necessary routines to accomplish the sort. The selection portion is shown in Figure 5.2. Essentially, it makes a list of those components to be sorted and indicates whether the components are to be chosen from above or below the mean for that component type. This list is called the sorting table; it is composed of two words per entry and is contained at the end of the subroutine.



STRT (See Figure 4.15)
DINS (See Figure 5.3)

Figure 5.2 Sorter Subroutine (Selection)

The sorting subroutine makes use of another subroutine called STRT, shown in Figure 4.15. This subroutine has the function of accepting a character inputted from the typewriter and examining it for four symbols, G, P, S, or CR (carriage return). A G, S, or P results in a return to the calling program with 0, 1, or 2 respectively in word TMP1. Should a CR be detected an exit made to the address stored in word OUT. If none of these symbols occurs, a slash is typed out and the carriage returned and a new symbol is awaited.

The sorting table is constructed in the following manner. The index word TBLI is initialized to the first location (TBLS), and the CR exit from STRT is set to the addres of SEX. The typewriter is caused to type out "SORT" and to execute a carriage return. At this point a call to STRT is made. If there are no elements to be sorted the operator presses the CR button, otherwise a G or S is typed in by the operator.

After this, the operator types in a string of integers representing the G or S number terminated by a plus (+) or minus (-). The plus indicates that the element is to have delays chosen from above the mean, while the minus signifies that the delays are chosen from below the mean. The routine checks the symbols, binarizing the G or S number until the plus or minus is sensed. An illegal

symbol results in a slash type out and a carriage return, after which a proper symbol is awaited.

When the terminating symbol is sensed, a pseudo label word is created. The binarized G or S number is shifted into bits 2 through 6 of this word and the contents of TMP1 are added to the word. This word is stored in the first address of the sorting table. The second address in the sorting table is set to 1 or 0 depending on whether an upper (1) or lower (0) sort is needed. A carriage return and line feed are made and control is transferred to STRT. If more elements are to be sorted, the process previously described is repeated. The sorting table is completed by pressing a CR button. This time the exit from STRT causes the next address in the sorting table to be labelled TBLF and control is transferred to the main simulation routine at GO (Figure 4.24).

The use of the sorting routine requires some change in the delay insertion procedure. The new version of DINS is shown in Figure 5.3. Once again the uniform density function is assumed for this example, although a change in assumed statistics would cause only minor modifications. The first portion consists of a search through the sorting table. For this search the matrix line label word is stripped of all extraneous information by retaining

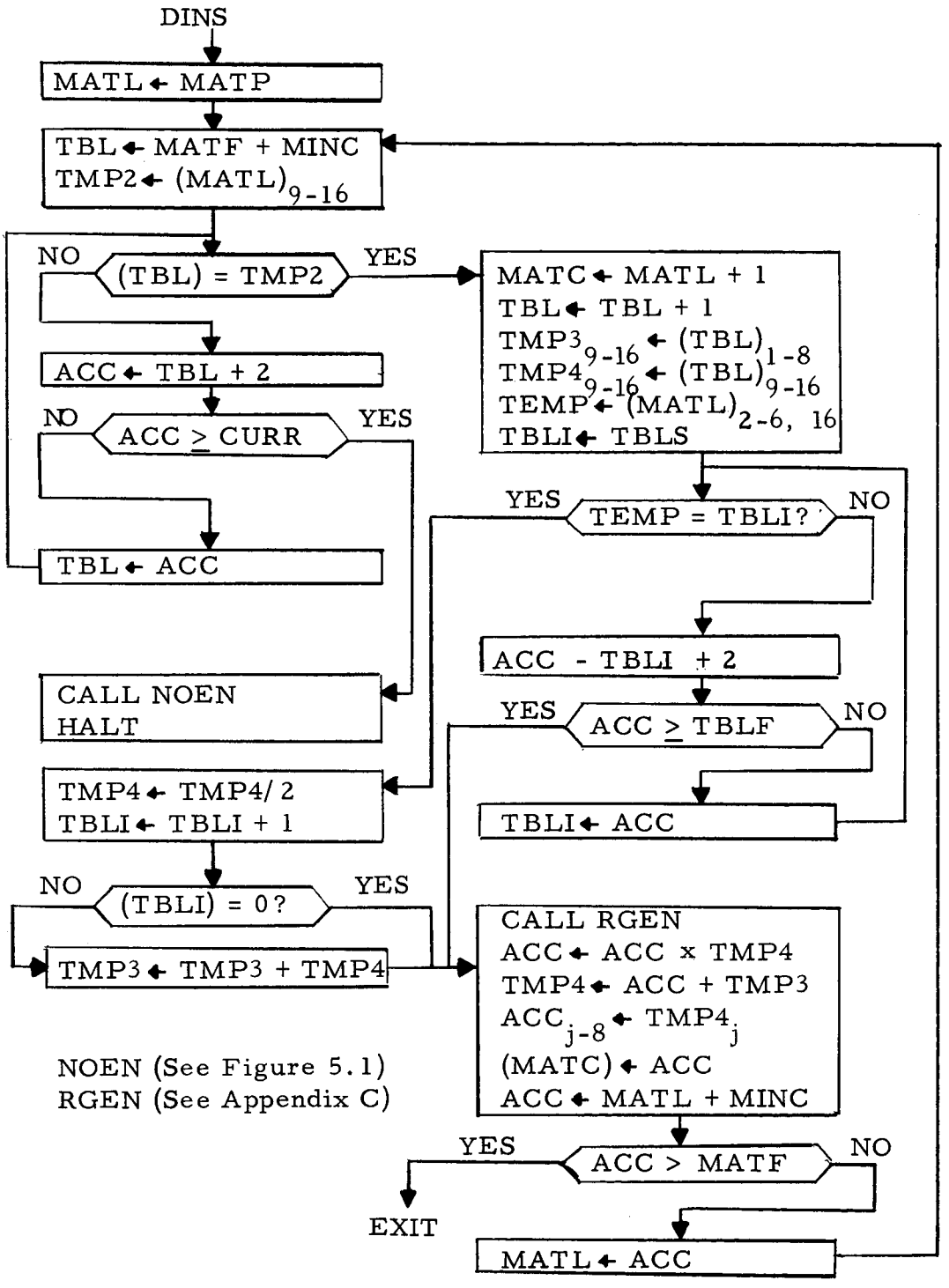


Figure 5.3 Sorter Subroutine (Insertion)

only bit 16 and bits 2 through 6 and then compared sequentially with all the pseudo labels in the sorting table.

If there is not entry in the sorting table the processing proceeds as previously described in the section on delay insertion. If there is a pseudo label word in the sorting table which matches the stripped matrix line label, then the word following the pseudo label is examined. This word determines the subsequent action.

If this word contains a 0, the density function width (W) stored now in TMP4 is halved. Thus the numbers produced by the random number generator are multiplied by W/2 prior to addition to B. On the other hand if the control word is a 1 denoting an upper sort, in addition to halving W, B is augmented by W/2. Then the remaining processing is as described in the preceding section.

5.4 The Flagging Routine

The selections of the transient in the system to be flagged is under the control of the user. At the time when flagging is to be accomplished (see the next section for an example) the ASR 33 types out "FLAG" followed by a carriage return and line feed (CR-LF). The operator then types in an A for an alpha or positive going transient or he types in a B for a beta, or negative

going transition. The operator then types in the P number, the G number, or the S number; terminating with a carriage return. The machine then executes a line feed and awaits a new signal to be flagged. If no further signals are to be flagged, an exit is made by a carriage return.

If symbols other than those discussed above are entered, the machine executes a CR-LF and awaits a legal symbol.

The flagging subroutine flow chart is contained in Figure 5.4 and 5.5. The first symbol typed in is examined to determine the nature of the transient. An A results in the storage of instruction SNZ (skip if not zero) in location SIGN. A B results in the storing of SZE (skip if zero). After this, subroutine STRT, discussed in the last section, is called in order to determine whether the flagged signal is on a P G, or S element. After this, the sequence of numerals typed in is binarized until a carriage return occurs.

The carriage return results in an examination of TMP1. It will be recalled that if STRT detected a P, G, or S, then TMP1 contains 2, 0, or 1 respectively. In the event TMP1 contains a 2 or 0, the location of the matrix line label word is computed by

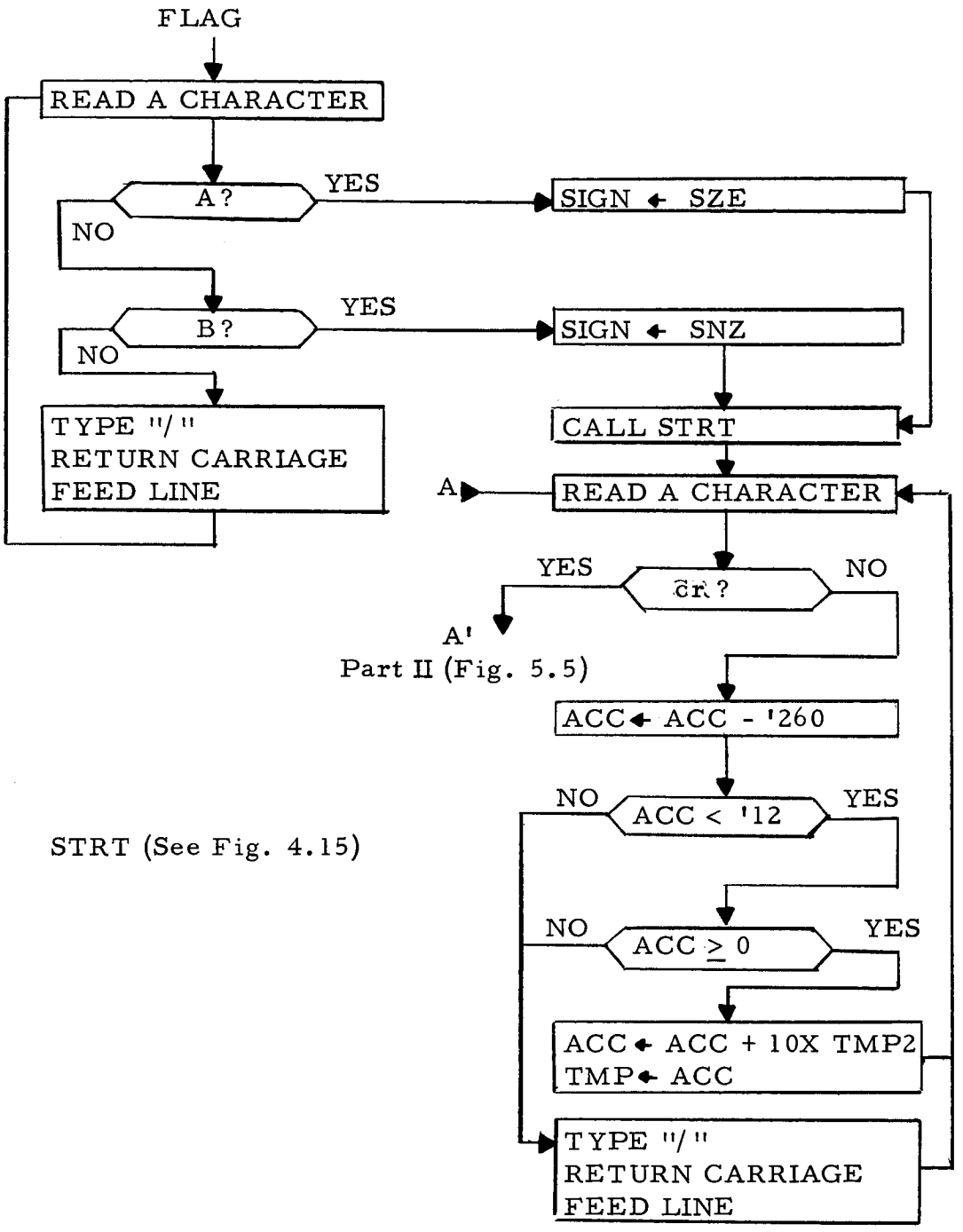


Figure 5.4 Flagger Routine (Part I) Along with Subroutine DTBI. DTBI is between Points A and A'.

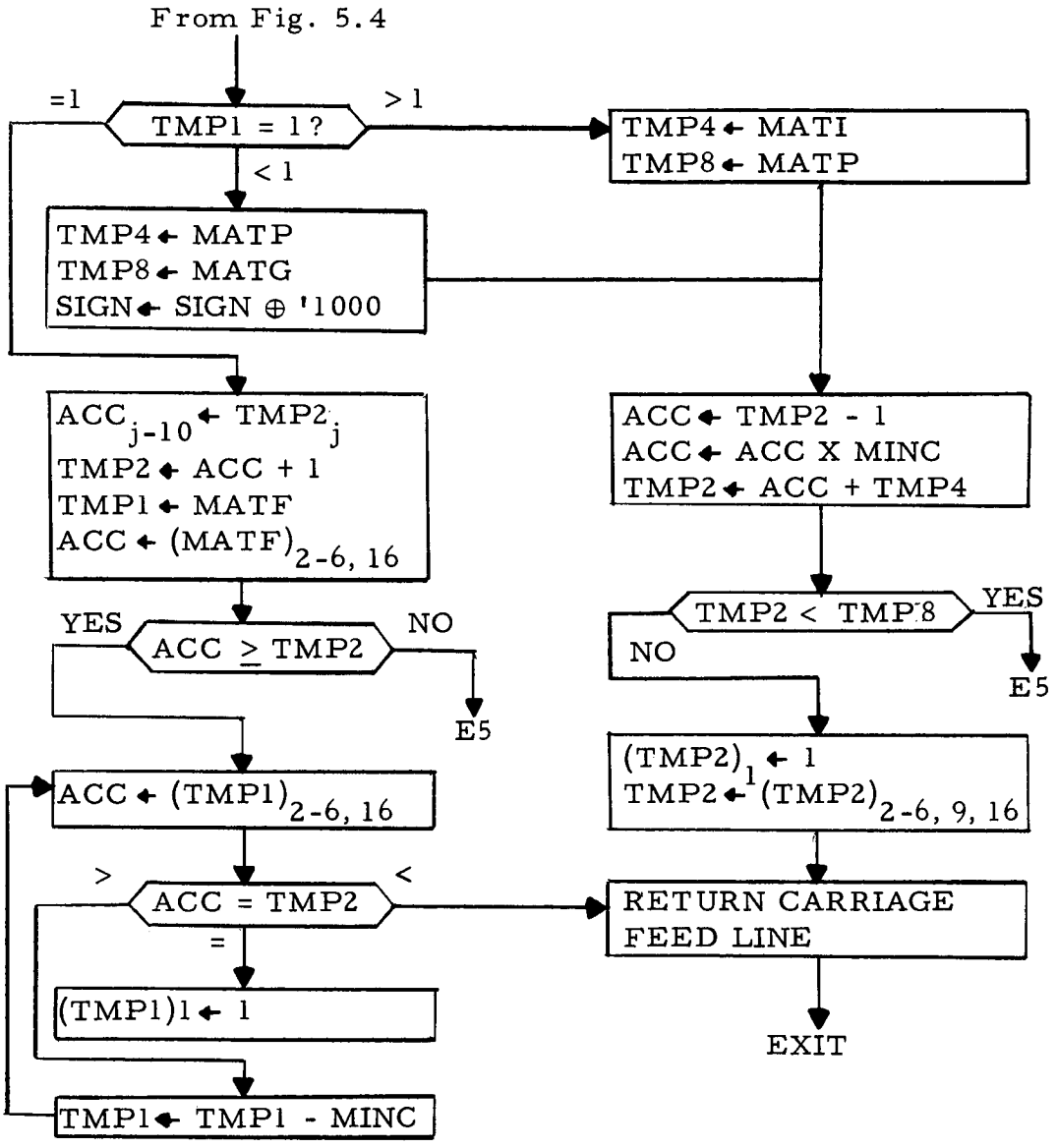


Figure 5.5 Flagger Routine (Part II)

multiplying the binarized P or G number by MINC and adding it to the address in MATI (P line) or that in MATP (G line). The flag bit in the computed address is then set to one. A check is also made here to determine if the computed matrix line label word lies in the proper group of matrix lines, i.e., an address computed for a P line must be between MATI and MATP. A computed address which does not lie in the proper interval results in a type out of E5 which indicates the address requested to be flagged is not a legal address.

Should TMP1 contain a 1, the element to be flagged is an S element. In this case a pseudo label word is created as was done in the case of the sorter routine. The line index is set to MATF and decremented until detection of lines whose S numbers are the same as the pseudo label. These matrix line label words are all flagged. In the event the S number requested to be flagged is greater than the S number contained in MATF, then an E5 is typed out. Following an E5 type out, the machine awaits a legal symbol input.

After the flagging is completed, an exit is made to the calling sequence with the address of the flagged element stored in TMP2.

5.5 The Statistical Analyzer.

As an example, the time difference between two transients will be considered.

This routine computes the time delay between alpha or beta transitions of two flagged elements. The parameters of interest in the statistical analysis are entered on the typewriter keyboard. The sequence of operations is as follows:

(a) The typewriter types out "ST", after which the user types in the lowest value of time which is of interest, and terminates by a carriage return.

(b) The typewriter types out "INT" after which the user types in the interval widths of his analysis; again termination is by a carriage return.

(c) The typewriter types out "NO" after which the user types in the number of the intervals in (b) which are required, termination is by carriage return.

(d) The typewriter types out "EV" after which the user types in the number of times he wishes the circuit to be simulated in order to gather statistics. Termination is by carriage return.

(e) The typewriter types out "FLAG" and "1st" after which the user types the nature of the transient (a or b) and the identifying number of the circuit element. The typewriter prints out "2nd" and the user presents the nature and identification of the second transient.

(f) A second carriage return begins the simulation, or if the sorter is to be utilized SORT is typed out and the sort is made as described in that section.

As a specific example, assume the user wishes to obtain a statistical analysis on the time delay in a system between an alpha transition on input signal P2 and a Beta transition on element G12. He then wishes to obtain information on the effect of sorting. Assume additionally that the user has no advance information on the spread of delays.

The user makes a crude estimate of the delay. For purposes of illustration, he guesses that the delay will lie between 0 and 500. To improve his estimate he divides this range into 50 intervals of 10 units each, and takes a sample of 300 simulations. When the program has been loaded, the following dialogue would take place at the ASR-33 keyboard. Underlined symbols are automatically typed out by the program, the others are entered by the

user:

ST 0(cr.)

INT 10(cr.)

NO 50 (cr.)

EV 300 (cr.)

FLAG

1st AP2 (cr.)

2nd BG12 (cr.)

SORT (cr.)

where

ST means the starting point of delay interval.

INT means the width of sub-intervals.

NO means the number of sub-intervals .

EV means the number of simulations .

(cr.) means a type in of a carriage return.

FLAG indicates the flagging routine has been called.

SORT indicates the sorter routine has been called.

Following this, the machine performs the simulation and types out 50 lines in two columns. Each line represents the contents of one of the intervals. The first column is the actual contents, the second column represents the accumulation of all the

preceding sub-intervals.

An examination of these results might yield that no delays in the 300 trials occurred less than 120 units or more than 320 units. If now desired a larger sample could be taken and the estimate improved. Now the dialogue might look like:

ST 100 (cr.)

INT 5 (cr.)

NO 50 (cr.)

EV 1000 (cr.)

FLAG

1st AP2 (cr.)

2nd BG 12 (cr.)

SORT (cr.)

Note that the interval of examination is from 100 to 350, in 50 sub-intervals of 5. If desired this process of adding precision to the statistics obtained could be continued as long as necessary. When the effect of sorting is to be tested, the elements to be sorted are entered after the machine type out of SORT. Since sorting lowers the variance, the same values of ST, INT, and NO as were used in the last unsorted simulations can be safely used.

As an example consider:

ST 110 (cr.)

INT 3 (cr.)

NO 70 (cr.)

EV 10000 (cr.)

FLAG

1st AP2 (cr)

2nd BG12 (cr.)

SORT

G4 + (cr.)

S2 - (cr.)

S4 + (cr.)

G7 - (cr.)

(cr.)

There is always the possibility that the second transient may occur before the first one in the system under test. Following the listing previously discussed, the word MISS is typed out followed by the number of times transients occurred in inverse order during the simulation.

The flow chart for the time difference routine is contained in Figures 5.6 through 5.8. The first part shown in Figure 5.6 consists of the initialization of some memory locations and the acceptance of data from the keyboard. The flow chart is almost self explanatory; however, some mnemonic definitions are in order:

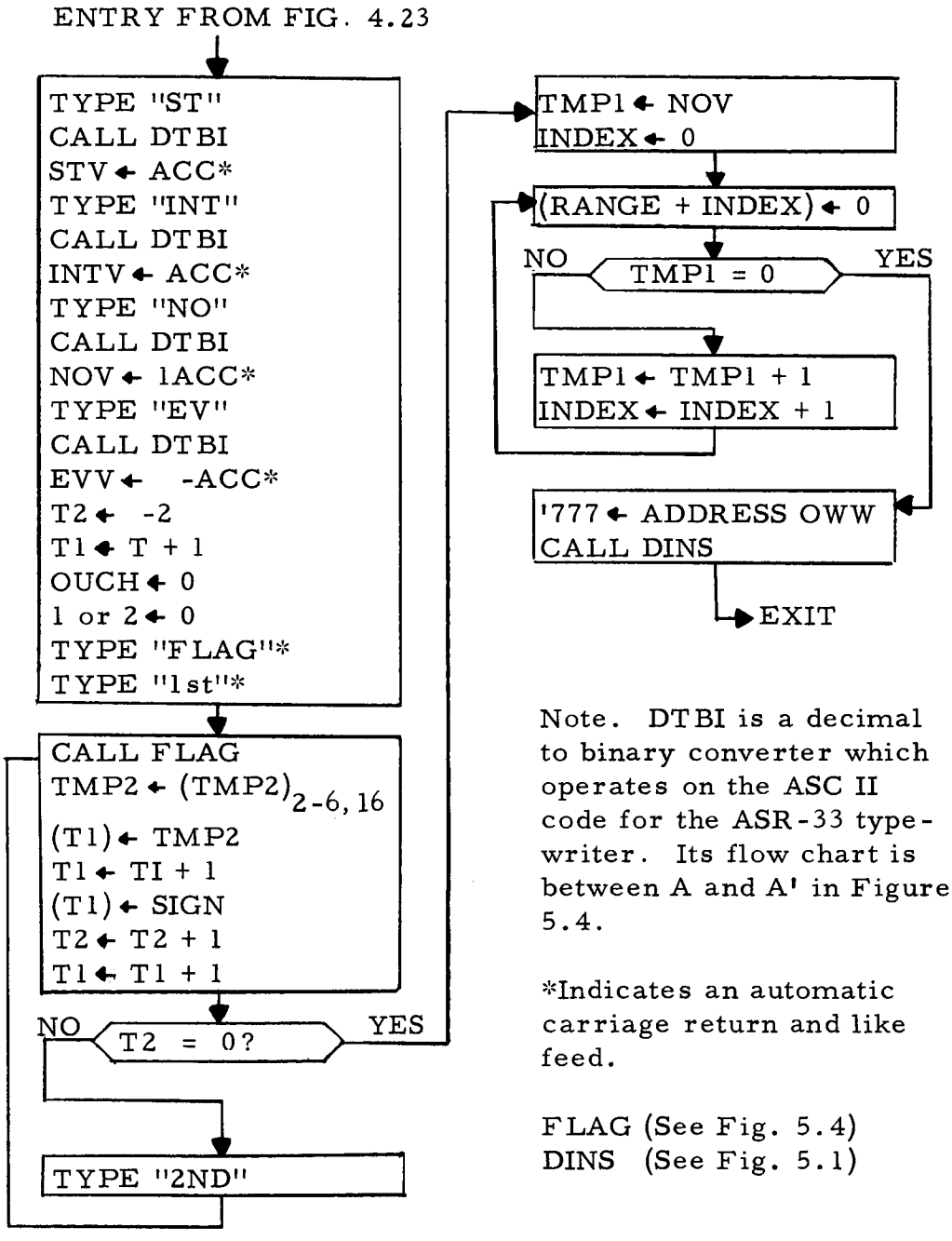
T2 is a control word which causes only two flagged inputs to be accepted.

T is the initial address where the label of the transients are stored alternately with their signs.

OUCH is the register which counts the MISSes discussed above.

1 or2 is a control word which is used in the determination of the order of the transients.

The end result of this portion of the routine is the clearing of a block of memory (RANGE) immediately following the program. The size of this block of memory is equal to the number of sub-intervals (NOV) in the analysis. Additionally the element type and number of the first transient are stored in T + 1 with its sign in T + 2. The element type and number of the second transient



Note. DTBI is a decimal to binary converter which operates on the ASC II code for the ASR-33 typewriter. Its flow chart is between A and A' in Figure 5.4.

*Indicates an automatic carriage return and like feed.

FLAG (See Fig. 5.4)
DINS (See Fig. 5.1)

Figure 5.6 Statistical Analyzer (Part I)

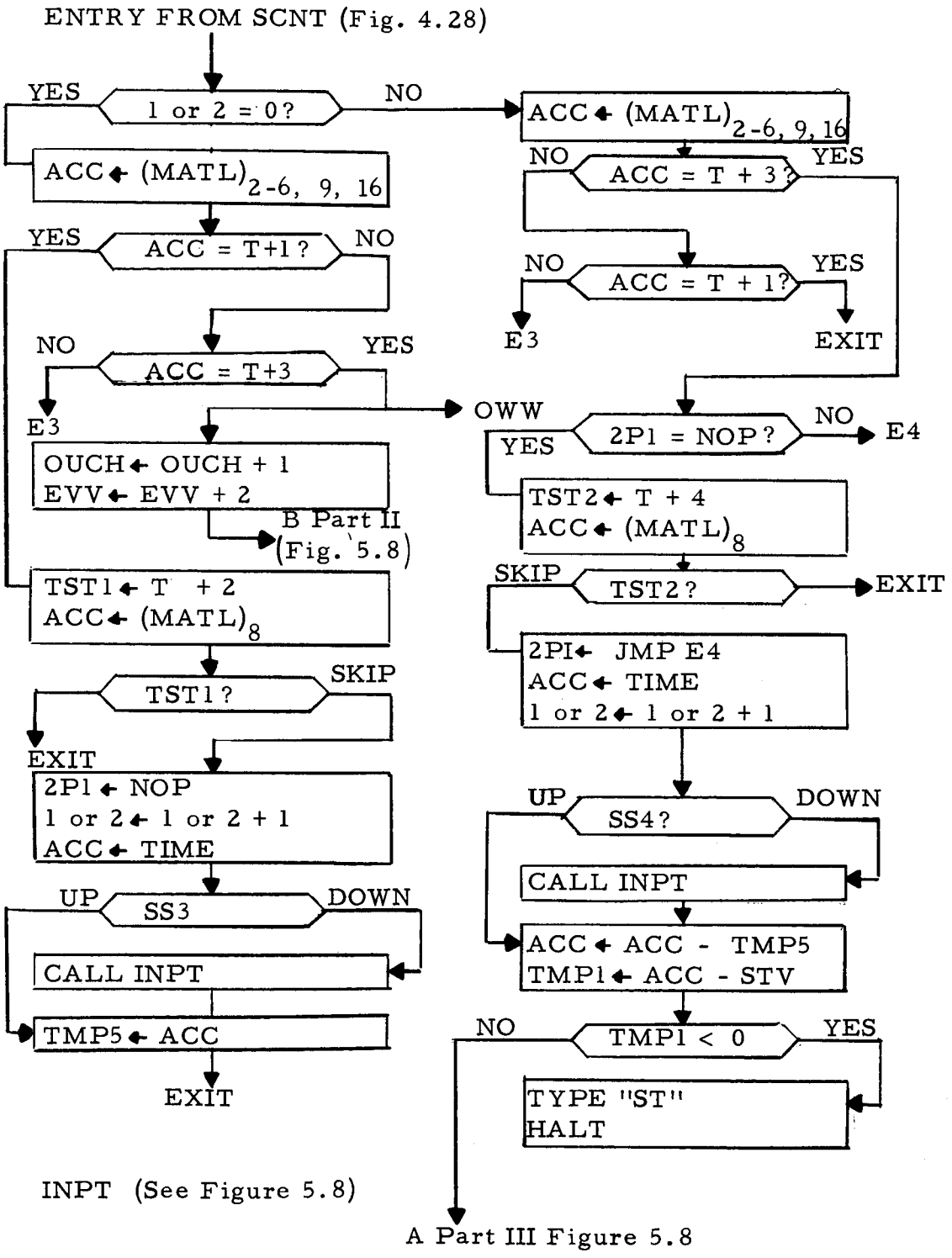


Figure 5.7 Statistical Analyzer (Part II)

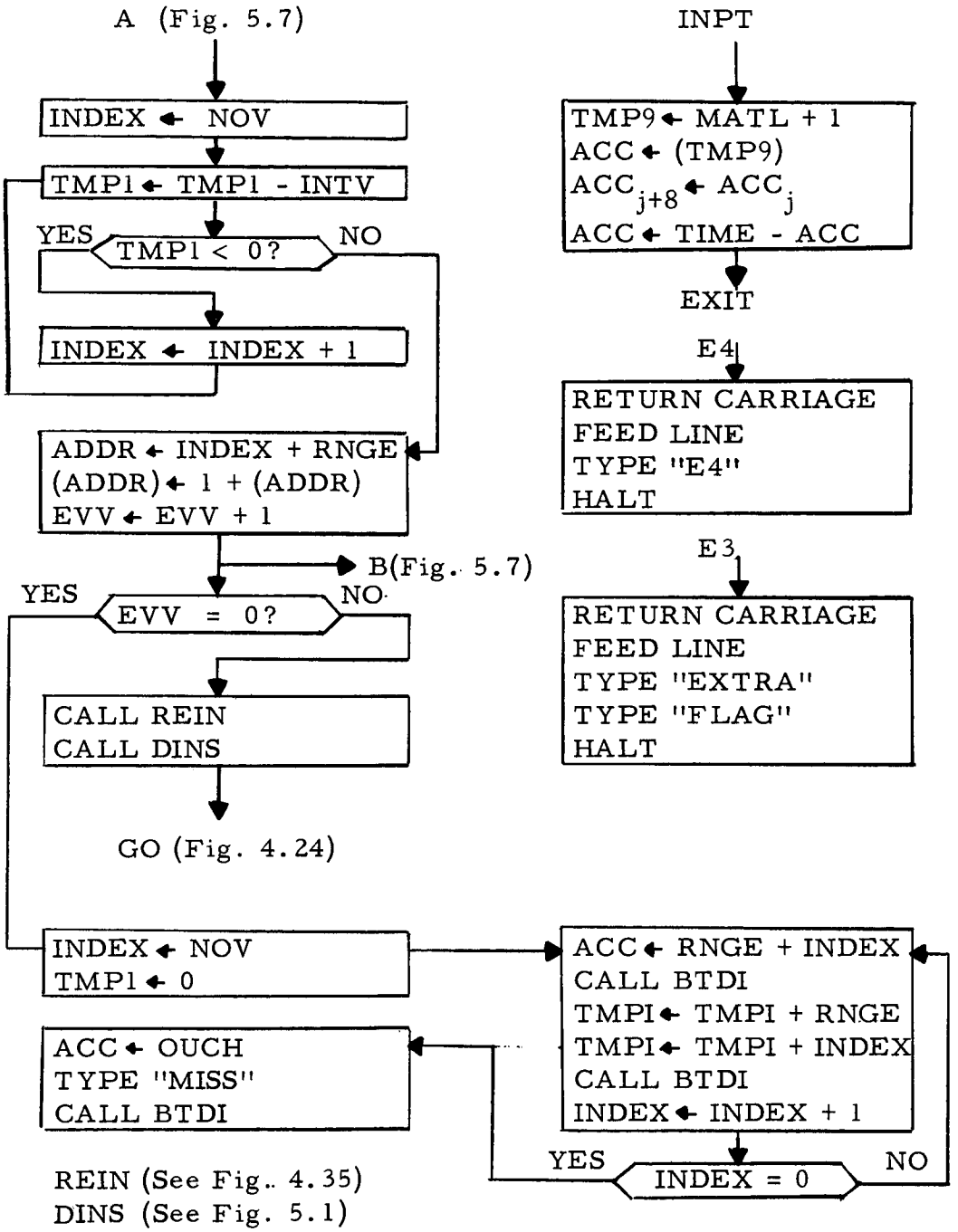


Figure 5.8 Statistical Analyzer (Part III)

are stored in T + 3 while its sign is held in T + 4.

The second portion, in Figures 5.7 and 5.8, shows the actual statistical analyzer in flow chart. Upon entry into the routine, the contents of location 10R2 are inspected. If it is even (i.e., bit 16=0) the processor has not processed a first transient as yet. Assume this is the case. The matrix line label word is stripped and compared in turn with the contents of T + 1 and T + 3.

If the stripped label agrees with the contents of T + 3, it means the transient expected second has arrived first. OUCH and EVV are incremented. EVV is tested to determine if the last event of the simulation has transpired. If not, the system is reinitialized, new delays chosen for the elements, and another simulation event occurs. If this were the last event the typewriter lists the analysis (this will be discussed in detail later).

If the stripped label is identical to the contents of T + 1, then the contents of T + 2 are inserted into TST1. It will be recalled that T + 2 contained an SNZ if the first transient were positive going, and an SZE is it were negative going. The status bit of the line label of the flagged element is subjected to the

TEST. If the status bit is 1 in the case of positive going transient or 0 in the case of a negative going transient processing continues otherwise an exit to the calling program is made. Further processing consists in the insertion of a NOP into control location 2P1, the transfer of the contents of TIME into TMP5, and the incrementation of 10R2 by 1 before an exit to the calling program.

If the stripped label did not agree with either T + 1 or T + 3, an exit to E3 is made. This causes EXTRA FLAG to be typed out and a halt executed.

If the contents of 1 or 2 were odd upon entry it means the second transient is expected. In this case, the stripped label is compared with T + 1 and T + 3. If it agrees with T + 1 a return is made to the main program. If it agrees with neither E3 is called. If it agrees with T + 3, it is indeed the expected second transient and undergoes further processing.

The matrix line label is checked for transient polarity using the contents of T + 4 as the test instruction (indicated as TST2 on the flow chart). If the polarity of the transient is correct 1 or 2 is incremented and the time difference is computed by subtracting TMP5 from TIME. The value of STV is subtracted from

the time difference and the result checked for a positive sign. If not, a halt is made after a type out of ST to indicate that the value of ST was poorly chosen. Operation can be continued if desired by pressing the start button.

However, if result of subtraction STV from the time difference is positive, the value of INTV is repeatedly subtracted from the result until the result goes negative. At each subtraction, the index register is incremented by 1, so that the proper interval is indicated by value of the index register. A 1 is then added into the address specified by RANGE plus INDEX, and EVV is incremented by 1. If this did not constitute the last event, the system is reinitialized, new delays are chosen, and a new simulation is begun.

The last event detection (EVV=0) causes the contents of each address in RANGE to be printed out along with the accumulations of preceding intervals. When all the sub-intervals are typed out, the contents of OUCH are printed following the symbols MISS, after which the machine halts, the analysis is completed.

A user's option has also been added. This option enables the user to select for measurement the time at which either

the beginning or the end of the propagation delay occurs. If sense switch number three is set (placed in the UP position) the end of the propagation delay on the first transient will be used, otherwise the start of the propagation delay. Sense switch four plays a similar role with respect to the second transient. This option allows the use of dummy gates to make otherwise difficult measurements. Its use is illustrated in the second example of Chapter VI. It should be noted that the option applies only to gates and does apply to storage elements.

CHAPTER VI

ILLUSTRATIVE EXAMPLES

6.1 General

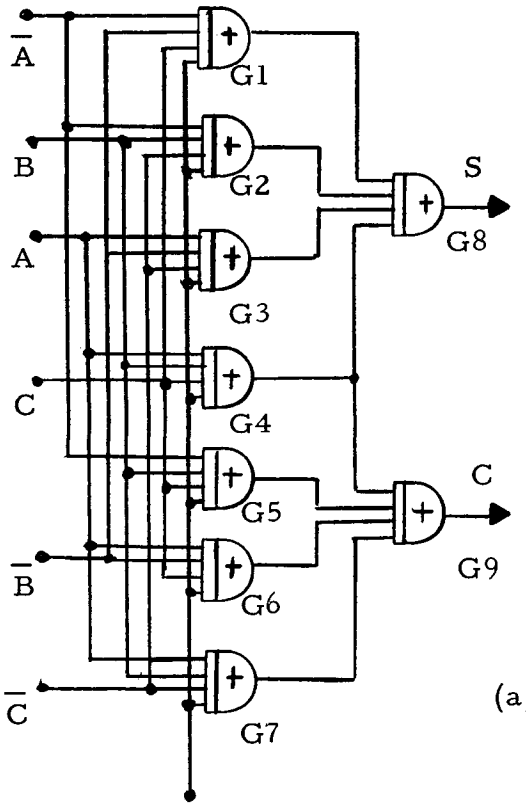
In order to demonstrate the effectiveness of the statistical analysis and the Monte Carlo simulation, several examples were run.

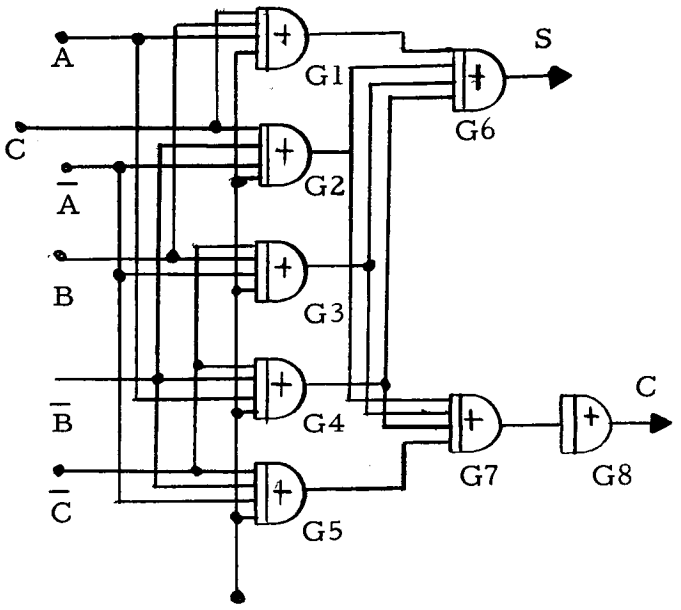
6.2 Adder Design

Figure 6.1 shows two possible full adder circuits using NOR circuits, taken from Ledley[18]. These are not chosen to illustrate optimum adder design but rather to show the influence of the statistical analysis on a design decision.

The circuit shown in Figure 6.1(a) is a standard two level NOR implementation. The alternate circuit in Figure 6.1(b) has been obtained by noting that the Boolean expression for $\overline{C}$ contains many terms common to those in the expression for S. It uses one less NOR circuit and requires the carry output to pass through three levels of logic.

If we consider the NOR gates in question to have a uniform spread in delays from 20 to 50 time units, a worst case analysis timing allowance indicates that the elimination of one





(a) Two level adder
9 circuits
44 solder joints
Input must drive 4 NORS

(b) Three level adder
8 circuits
37 solder joints
Input must drive 3 NORS

$$S = ABC + A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C$$

$$C = ABC + \bar{A}BC + A\bar{B}C + AB\bar{C}$$

Figure 6.1 Two Binary Adders

circuit is obtained at a price of a 50% increase in propagation delay.

A statistical analysis of the response of three gates in sequence using 40, 000 simulations, however, indicates that the probability that the delay will exceed an increase of 40% is less than 10^{-4}.

The designer may then investigate the effects of component sorting. The eight circuits of Figure 6.1(b) are sorted as follows. Circuits G1, G2, G3, and G4 contain the upper sort or those drawn from the sort above the mean; while G5, G6, G7, and filled with components drawn from the lower sort. The delay densities for the carry output obtained by simulation are shown in Figure 6.2, where it is compared with the results for the adder of Figure 6.1(a) unsorted. Note that the predicted 50% increase is now less than 20%.

Figure 6.3 shows the comparison of the carry output of the three-level adder for the sorted and unsorted case.

Finally, Figure 6.4 shows the two-level adder unsorted compared with a sort on the two-level adder. Elements G1, G8, G9, and G7 come from the lower sort, while G2, G3, G5, and G6

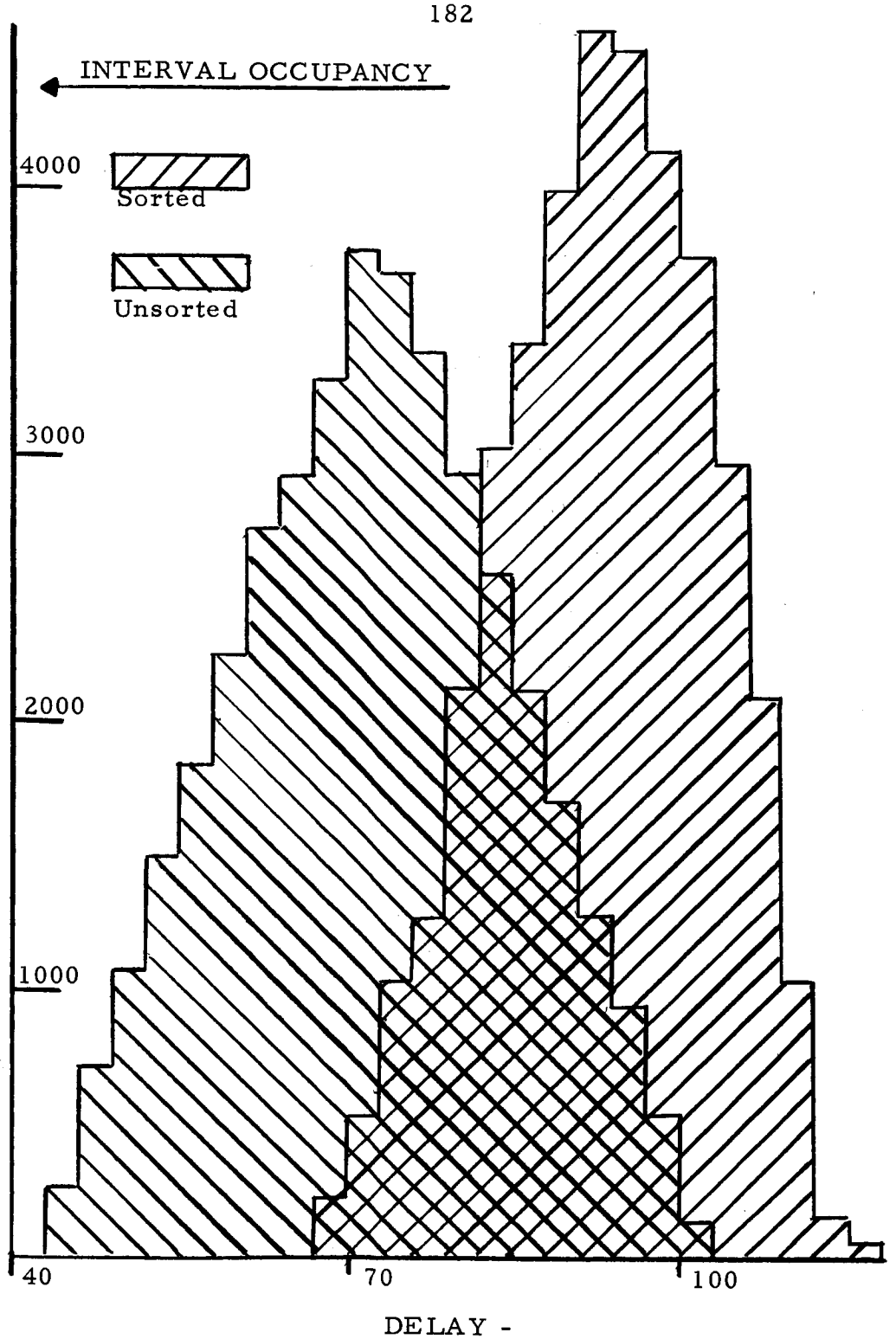


Figure 6.2 Comparison of C Outputs Between Two Level Circuit (Unsorted) and Three Level Circuit (Sorted).

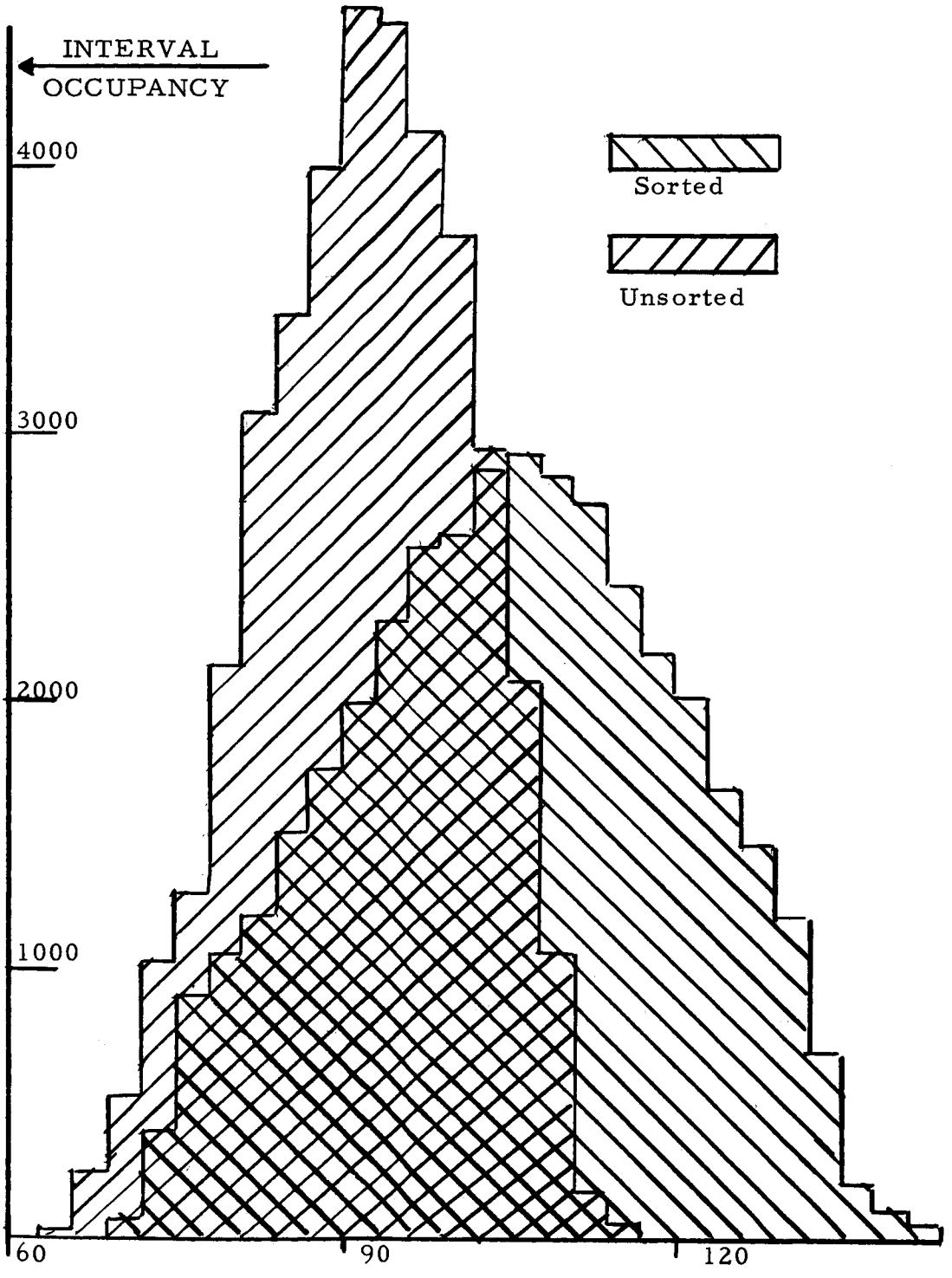


Figure 6.3 Comparison of Carry Outputs With and Without Sorting for the Three Level Circuit

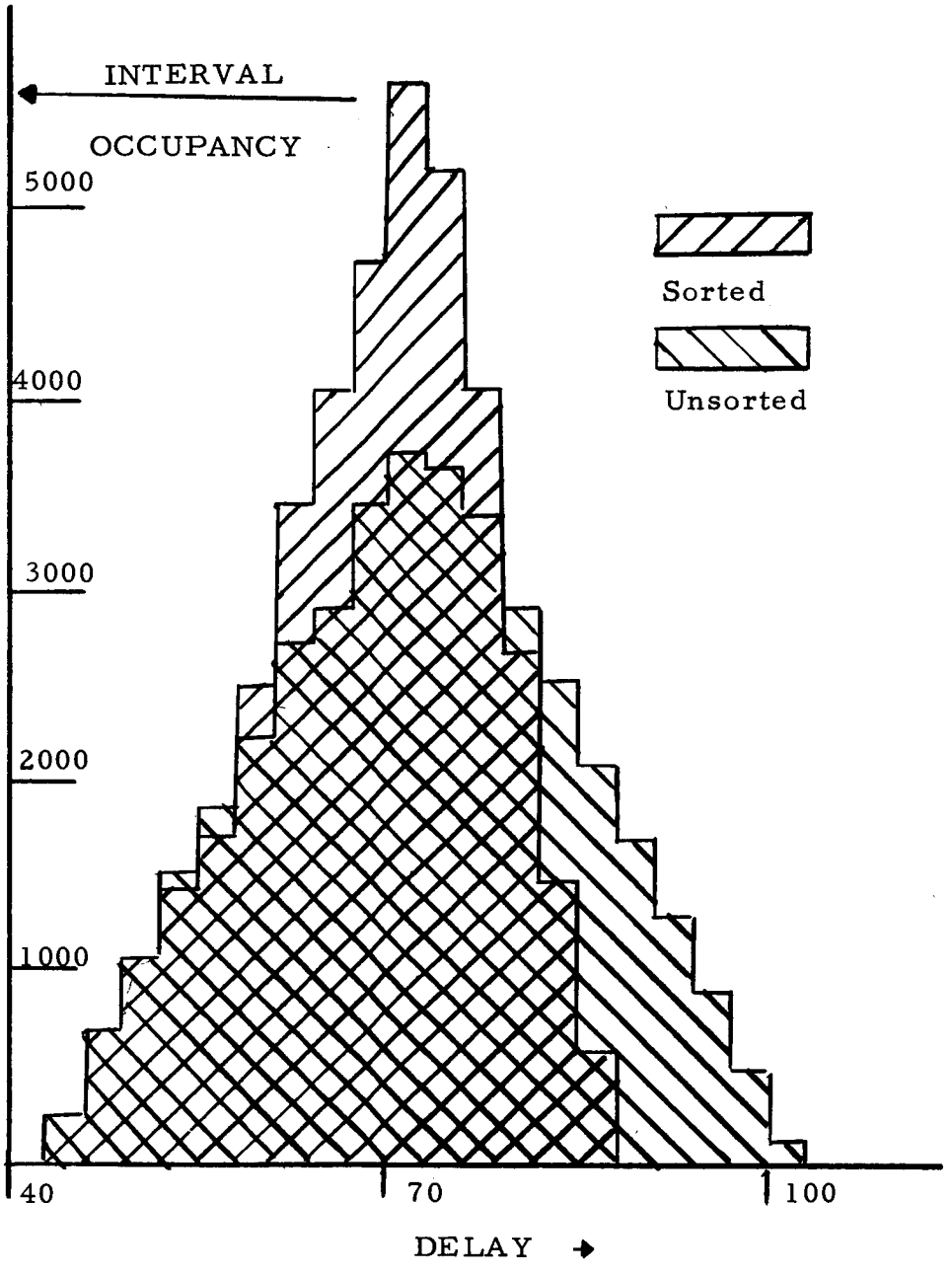


Figure 6.4 Effect of Sorting on the Two Level Adder

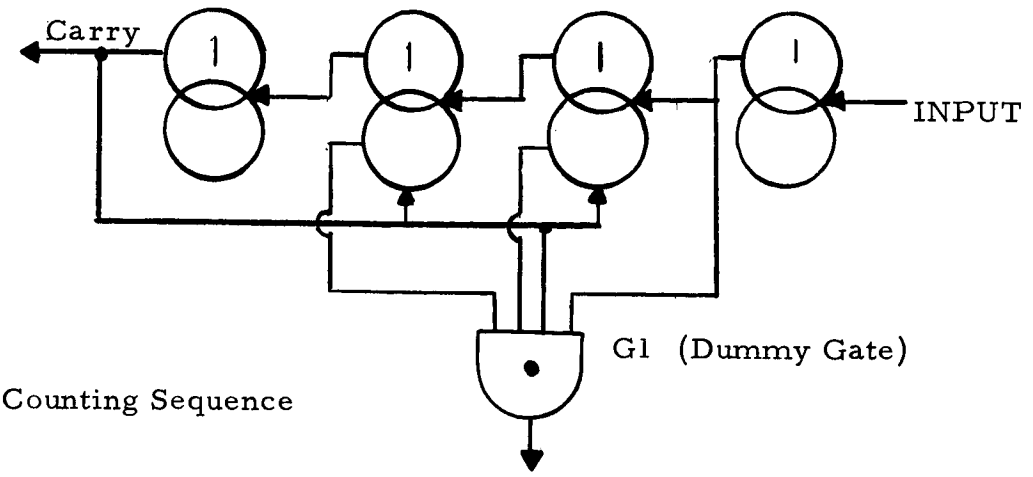
come from the upper sort; G4 is unsorted.

The designer can now make a decision between the two adders based on the economics of component price, cost of sorting and cost of time delay using information not available under worst case conditions.

The time necessary to compile the statistical information was about 15 minutes most of which was consumed by system and preparation, and by typewriter print out of data.

6.3 Feedback Counter Ripple Time

Figure 6.5 shows a decade of a decrementing binary coded decimal counter. The counting sequence is shown below in the same figure. Stable states of the counter are underlined while transitory states which occur between legitimate counts are not underlined. Two important considerations in counter design are the time taken to propagate a carry and the maximum time elapsing between stable states. The carry delay is the time between the arrival of a pulse to be counted (when the counter is at all zeroes) and the positive going transition on S4. A glance at the system counting sequence shows that maximum time between legal counts also occurs at this point in the counting sequence.



Counting Sequence

9	1 0 0 1
8	1 0 0 0
	1 0 0 1
	1 0 1 1
	1 1 1 1
7	0 1 1 1
6	0 1 1 0
	0 1 1 1
5	0 1 0 1
4	0 1 0 0
	0 1 0 1
	0 1 1 1
3	0 0 1 1
2	0 0 1 0
	0 0 1 1
1	0 0 0 1
0	0 0 0 0
	0 0 0 1
	0 0 1 1
	0 1 1 1
	1 1 1 1
	1 0 1 1 or 1 1 0 1
9	1 0 0 1

Figure 6.5 Decrementing Decimal Counter

Note that there are five intermediate states in the 0 to 9 transition and a maximum of three intermediate states elsewhere.

The statistical measurements were taken using the "Time Between Transients Routine" discussed previously (in Section 5.5). A dummy gate, G1, was connected to recognize the count of 9, while the counter was initially set to all zeroes. The routine was required to measure the time between positive transition on the output S1 and the time the input on G1 initiated a positive going transition on its output.

It was not obvious where the delay time was located, so a trial run of 300 simulations was made using the delay range from 0 to 500, divided into 50 sub-intervals. These results causes a second trial of 500 events to be attempted with the delay range starting at 100 and using 50 sub-intervals of 5 each to extend the range to 350 units. These results indicated a further narrowing of the range was possible. A run of 1, 000 events was made, starting at 130 and using 50 sub-intervals of 3 each. This trial had results which indicated that further narrowing of the range was not desirable. This process of "tying down" the range of delays occuppied approximately one minute of computer time. However, the relatively slow speed of the ASR-33 print-out caused the total

time involved to be approximately 10 minutes.

It was decided to make two runs of 10,000 events each. The first run would have S element switching delays drawn from an unsorted uniform density function between 20 and 50. In the second run, S3 and S2 were drawn from the lower sort while S1 and S4 were drawn from the upper sort. The choice of elements into sorts is rather easy this time since S2 and S3 are switched twice in the transition from 0 to 9. The carry is unaffected by the choice since each of the S elements switch once in producing the carry and, as it was indicated in Chapter I, the order of insertion of the sorted elements is immatrial.

It is in order, at this point, to note that the statistical data on the transfer from 0 to 9 could not be obtained by standard convolution techniques since the time to go from 1111 to 1001 depends on the longest switching time between S3 and S4. This is analogous to the AND gate discussed in Chapter III.

The statistics on the delay between 0 and 9 are shown in Figure 6.6 for the sorted and the unsorted case. The designer can then make a decision as to the economic desireability of the sorting. Note that the mean is lowered by the act of sorting here.

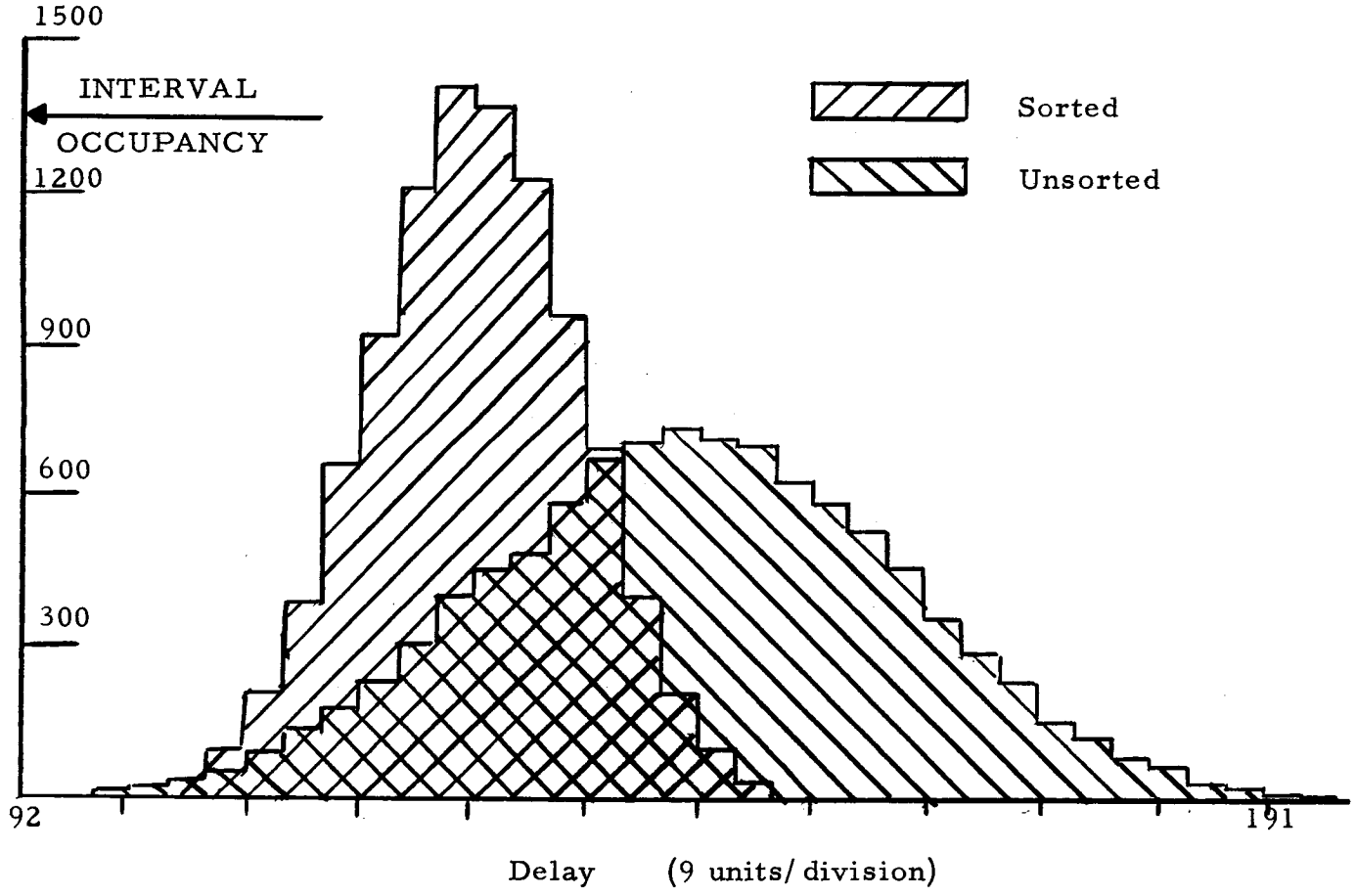


Figure 6.6 Time Between "0" and "9" for Sorted and Unsorted Cases 10, 000 Trials. Total Machine Time 7:33. The Worst Case Time Would be 250 Units.

This is due to the fact that, as previously mentioned, the delays due to S2 and S3 occur twice while those due to S1 and S4 occur once.

It is worth noting here that the correlation between the resetting speed (R input delay) and toggling speed (T input delay) was handled by choosing them both from the same sort of the delay statistics. It could well be argued that for some physical RST elements, a more realistic model would have been to use the same delay for each element. The effect is to broaden slightly the delay density function of the sorted elements.

The results of the statistical analysis on the carry output are shown in Figure 6.7. As would have been expected from the material of Chapter II, the mean delay stays the same, while the range of delays is shrunk to one-half.

6.4 Race Detection

Many times when building digital systems from micro-circuit components it is convenient to use the inherent delays in the elements to ensure a proper sequencing of signals at another portion of the system. As an example consider chains A and B of Figure 6.8. Chain A contains 6 elements while chain B contains

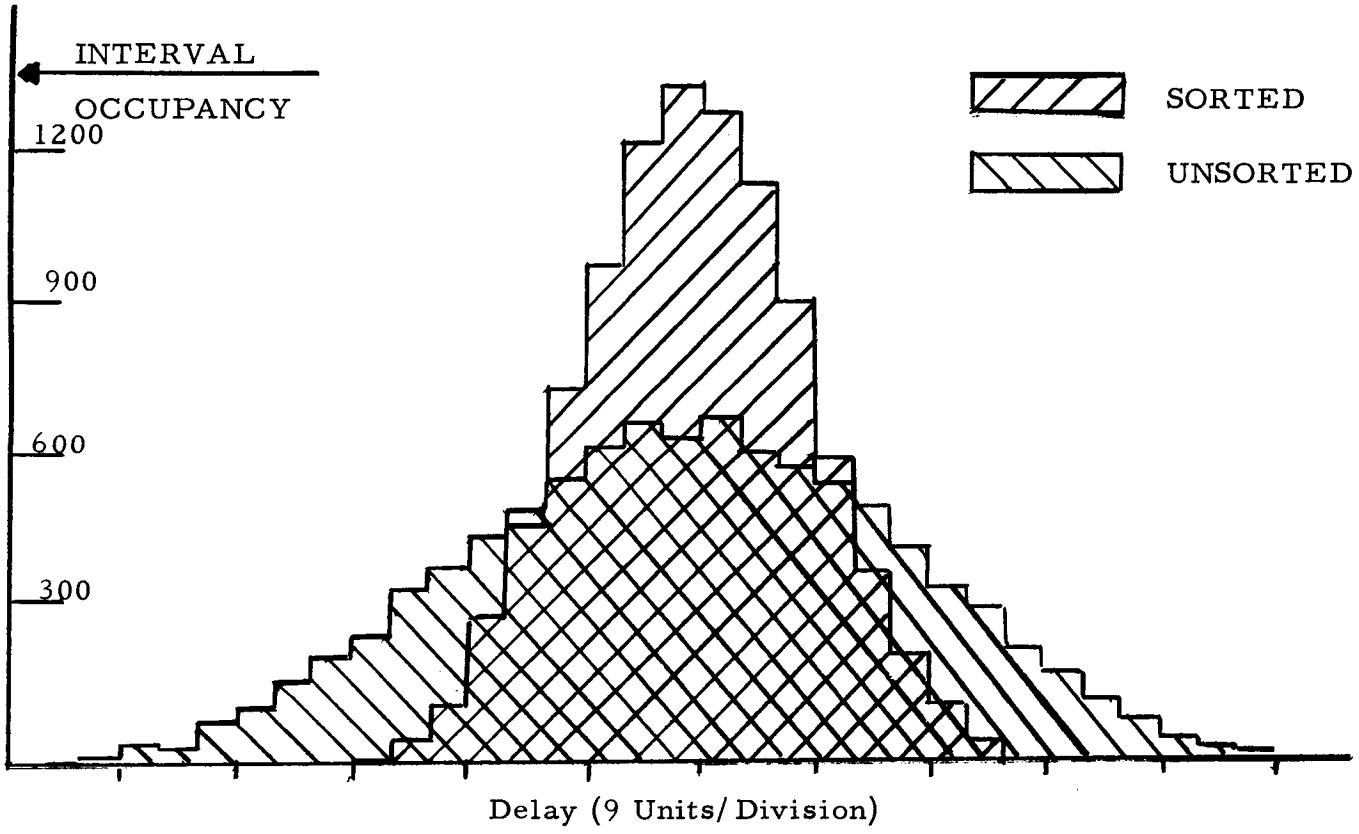
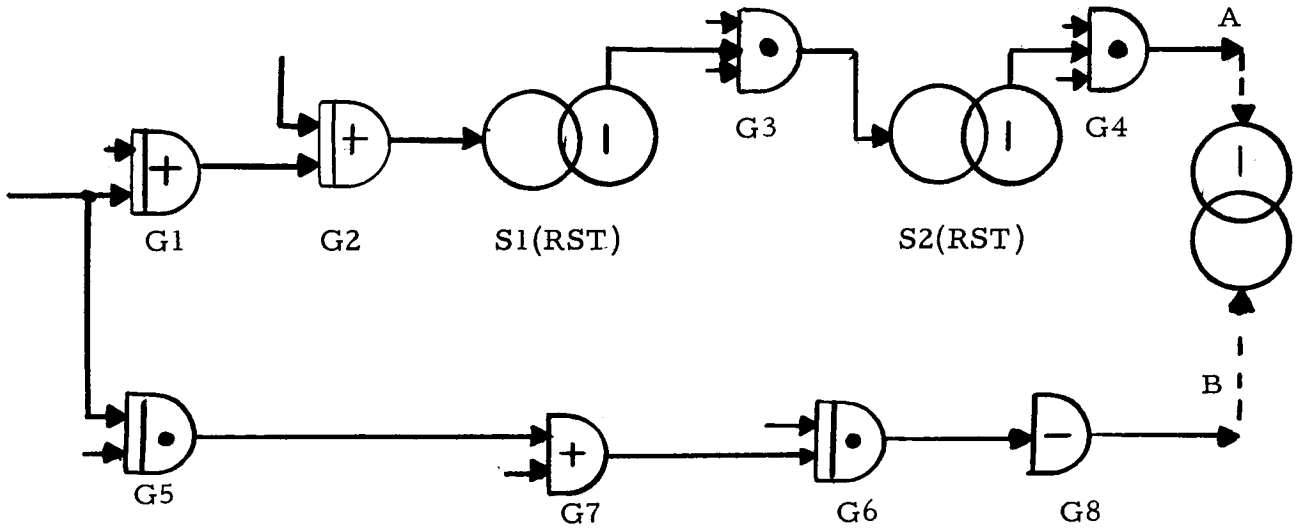


Figure 6.7 Carry Ripple Time for the Sorted and Unsorted Cases 10, 000 Trials. The Worst Case Time Would be 200 Units.



It is assumed that the inputs shown always determine the start of propagation in the gates.

Figure 6.8 Two Logic Chains Used to Ensure Delay

only 4. It is hoped by the designer that the two extra element delays of chain A will ensure that the signal at the output of chain A will always arrive after the output signal of chain B.

A worst case analysis using the minimum delays in chain A and the maximum delays in chain B indicates that overlap is possible. The designer would like the answer to two questions:

(a) Is the probability that the output of chain A will arrive first significant?

(b) Can the sorting operation be used to improve things?

The first question is answered by statistically analyzing the time difference between the output of chain A and the output of chain B. The results are shown in Figure 6.9. Also shown in Figure 6.9 are the results for the following sorts:

(a) Chain A

Upper Sort G1, G3, S1

Lower Sort G2, G4, S2

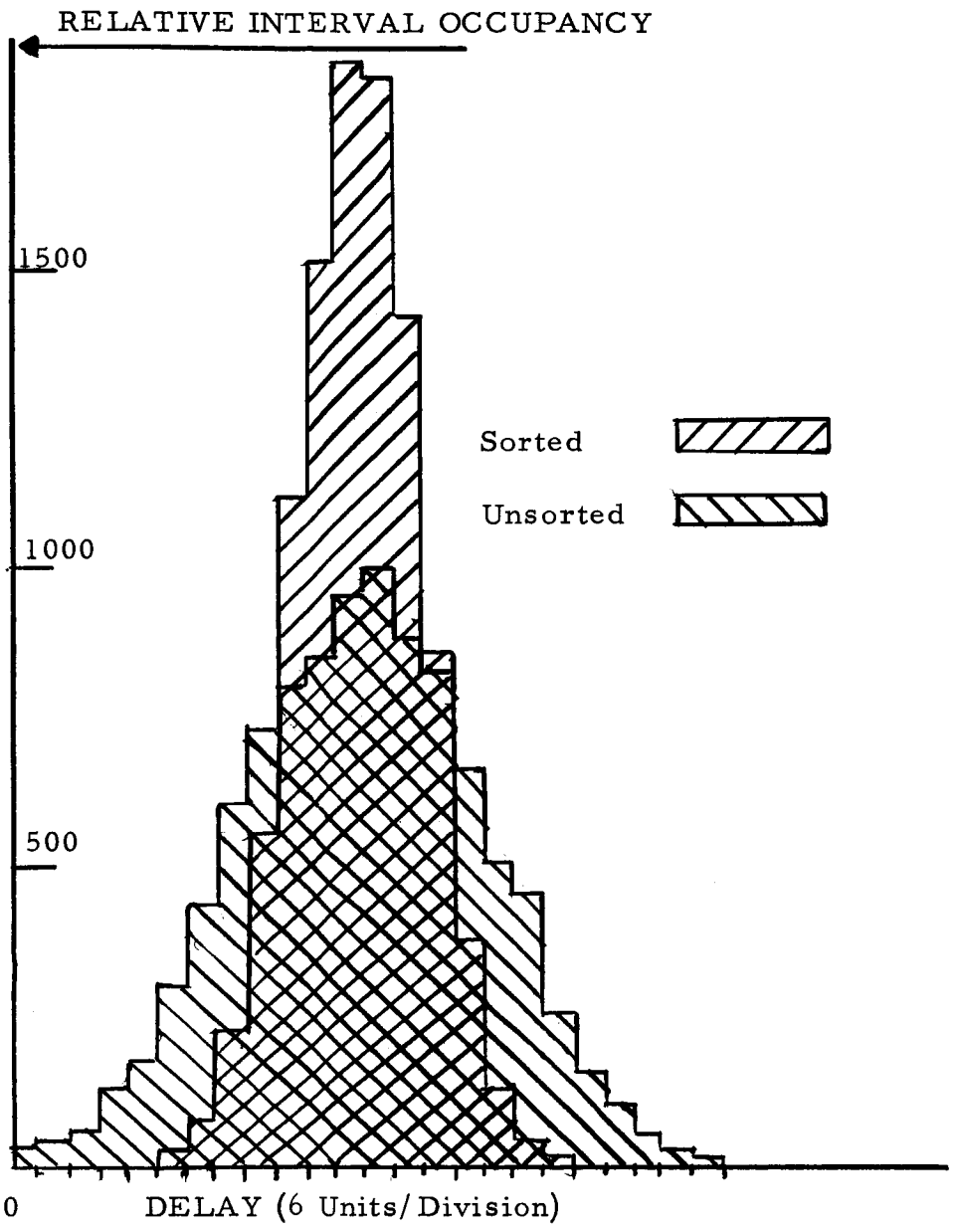


Figure 6.9 Delay From a G8 to a G4 10,000 Trials
The a G4 Occurred first 14 Times
Without sorting, never with sorting.
Total Computational Time 7:20

(b) Chain B

Upper Sort G5, G7

Lower Sort G6, G8

Note that the sole act of sorting the components has eliminated the possibility of the output of chain A arriving ahead of the output of chain B.

This example is rather artificial since no possibility exists for sorting elements such that the upper sorts go into chain A while the lower sorts of the same element type go into chain B. This latter sort in addition to changing the race possibility would also change the mean delay in each chain, lowering it is chain B and raising it in chain A.

Figure 6.10 shows the results of a more practical case. The two chains analyzed above are used to create a pulse by setting and resetting an RST flip-flop. A flip-flop was included in the simulation and the statistics were gathered on the width of the pulse generated.

It will be recalled that the ground rules for the RST flip-flop cause no pulse to be generated if the resetting transient

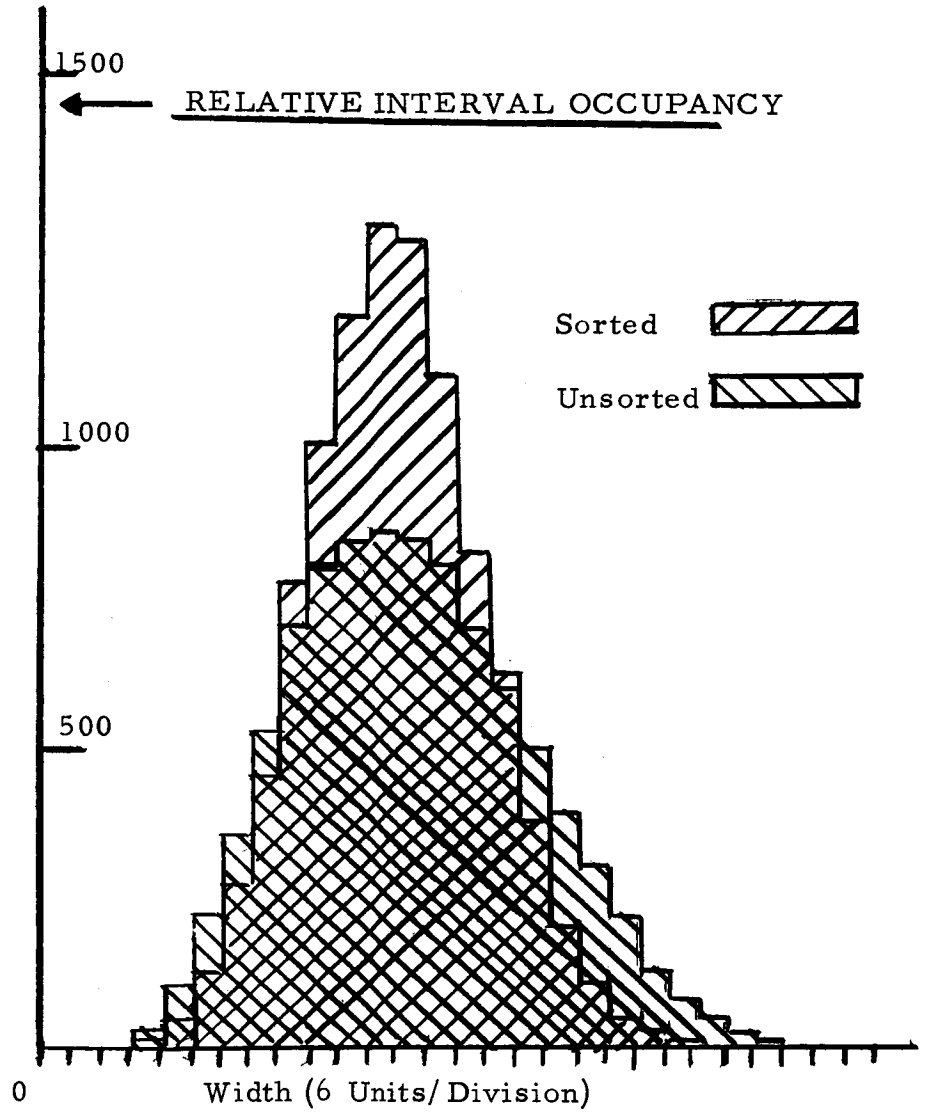


Figure 6.10 Width of S3 Output Pulses
10,000 Trials
No Pulse Occurred 70 Times with Sorting, 867 Times Without Sorting
Total Computational Time 8:40

arrives during the propagation time of the setting transient. In this case then the outputs of chains A and B must be separated in time by an ammount at least equal to the propagation delay of the flip-flop which is, itself, a random variable.

To implement this, the no output condition of the simulation was caused to make an entry into the MISS counter, by setting the address of location OWW into control word '777.

In this more practical case, the results shown in Figure 6.10 indicate that sorting does not eliminate races but does reduce their incidence by a factor of more than ten for the example chosen. In this example the delays were again assumed uniform as follows:

AND	15 to 35
OR	15 to 35
NAND	20 to 50
NOR	20 to 50
RS	20 to 50

CHAPTER VII

SIMULATION SPEED

7.1 General

There are many factors which influence the speed with which statistical analyses can be obtained. For this reason, a high degree of accuracy in predicting speed is not possible; however, many qualitative statements can be made.

The factors influencing speed can be grouped into three categories:

(a) the statistical processing

(b) the delay insertion and reinitialization

(c) the simulation programs.

Note that the time taken to prepare the matrix will not be considered in detail. The speed here is limited by the speed of the card reader and can be obtained by dividing the number of cards, T through E, by the reading rate of 100 cards per minute.

Nor will the time taken to enter data at the ASR-33 keyboard be considered since this time is primarily determined by the reaction time of the user and only secondarily by the response time of the ASR-33.

7.2 Statistical Processing

Quite clearly, the statistical processing time will depend on the nature of the statistics involved in the system. For purposes of illustration here, the "time between transients" routine described in Chapter VI will be discussed. Various computational times of interest are listed in Table 7.1. One fact which stands out is that the statistical analysis time is quite largely determined by the mean number of intervals traversed before the proper interval for the measurement is found. The mean number of intervals tranersed will depend on the nature of the statistics. Consider, for example, the three cases shown in Figure 7.1. Case (a) is a symmetric density function, while (b) and (c) are lopsided to the right and left respectively and have means which are correspondingly higher and lower than the mean of case (a).

If we make the following assumptions, a numerical example can be posed.

(1) No entries into MISS .

(2) First transient is an output and has the right sign the first time.

Table 7-1

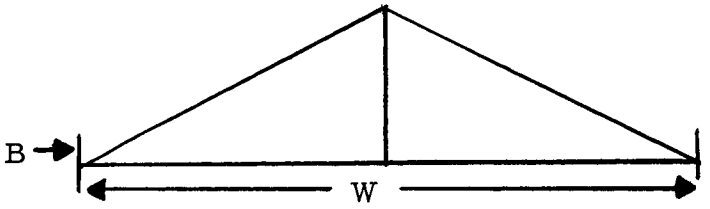
Computational Times for the Time Between Transients Processor

A. Time to process an entry into MISS	56.1 μ sec to 57.8μ sec
B. Time to process the first transient	
1) if wrong sign	42.5 μ sec
2) output transient, right sign	66.3 μ sec
3) input transient, right sign	98.6 μ sec
C. Times involved in second transient processing	
1) if first transient signal exit	28.9μsec to 30.6 μ sec
2) if wrong sign	45.9 μ sec
3) output transient, right sign	125.8μ sec + 28.9(X-1)μsec
4) input transient, right sign	158.1μsec+28.9(X-1) μ sec

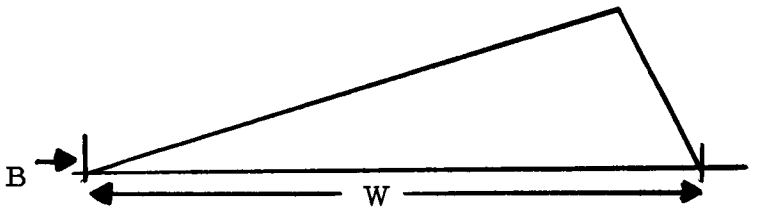
* X is the number of the interval into which the measurement falls.

(a) Symmetric

Mean = $\frac{W}{2}$



(b) Mean = $\frac{3W}{4}$



(c) Mean = $\frac{W}{4}$

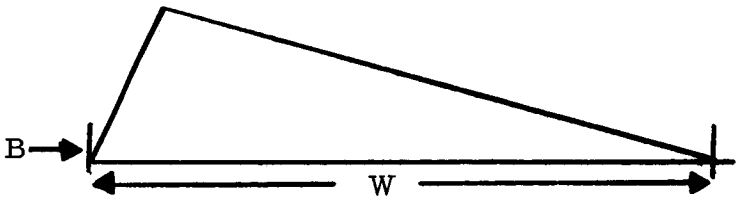


Figure 7.1 Assumed Output Statistics

(3) The next signal processed is the second transient, an input transient, with the right time.

(4) The range of delays is broken into fifty intervals.

(5) The total number of simulations is 10,000.

Referring to Figure 7.1 , the total time involved in the statistical analysis, for case (a) is 9.18 seconds. For case (b) it is 12.94 seconds, while for case (c) it is only 6.00 seconds.

7.3 Delay Insertion

Once again, the delay insertion time is governed by the nature of the statistics, only this time it is the statistics of delays of the elements. Computational times of interest are shown in Table 7.2 . Here a salient factor is the effect of the position in the delay table of the element type as discussed in part D. It bears out the earlier statement that time can be saved in processing if the most common elements in the system are placed early in the delay table. Recall that the order of placement of element types in the delay table is at the option of the user.

To obtain an order of magnitude for the time involved a complex digital sub-system, consider the following assumptions:

Table 7.2

Computational Times for the Delay Insertion Routine

A. Random number generator	73.4 μ sec
B. Generation of a number to a uniform density function (including A)	126.7 μ sec
C. Generation of a number to a triangular density function (including A)	267.0 μ sec
D. Time required to generate and insert a delay* governed by a uniform density function, where E' is the position of the element type in the delay table.	226.4+28.9(E'-1)μsec
E. Additional Time required for sorting	
1. Elements not in the sort	75.0+28.0 N" μ sec
2. Elements in the upper sort	95.4+28.0(E"-1)μ sec
3. Elements in the lower sort	85.6+28.0(E"-1)μ sec
where E" is the position of the element in the sorting table.	

* The first matrix lines require 8.5 μ sec additional, and the last line 3.4 μ sec less.

(1) The circuit contains 5 signals, 20 G elements and 20 S elements.

(2) Each S element requires two matrix lines and the two inputs are assumed independent.

(3) Six elements (4 G and 2S) are to be sorted.

(4) The delays in question are governed by a uniform density function.

(5) There are five element types (4G and 1S) used in the system. The S element is the first entry in the delay table and for an S element E'=2 on the average. Also the gates types are intelligently placed in the table so that for a gate E'=5. S elements are also placed first in sorting table.

(6) There are to be 10, 000 realizations of the circuit.

The total time taken then for insertion of delay is approximately 170 seconds, about two-thirds of which is occupied

with processing the S elements. Thus an increase in the number of S elements has about twice the effect on delay insertion time as a numerically equal increase in the number of gates.

It is noticed that the use of sorting requires 88 additional seconds over the 148 seconds required for delay insertion with no sorting.

In addition, note that P lines are not acted on by the delay insertion routine. This fact should discourage the use of flip-flops and gates which provide fixed levels during the time of the system transient under study. These elements should be replaced by fixed level P signal instead.

7.4 Reinitialization

Each time a new event in the simulation is begun, the circuit must be returned to its initial conditions. Computational times of interest are shown in Table 7. 3. Note that an S element with, say, two matrix lines requires 241 micro seconds which is slightly more than double the time required for reinitialization of a G or P element.

Table 7.3

Computational Times for the Reinitializer

A.	Entry, initialization of constants, and exiting	98.6 μ sec
B.	P line processing time*	112.5 μ sec
C.	G line processing time*	
	1) No change required	83.2 μ sec
	2) Change required (average)	98.0 μ sec
D.	S element processing time	
	1) Test for proper bit (R or S) fails	64.6 μ sec
	2) Initialization and exit	88.0 μ sec
	3) S line processing (per line)	44.2 μ sec
E.	Time to move to next word of INIT	11.9 μ sec

*Subtract 6.8 μ sec for the first line of the type.

If the assumption that one-half of the G line require changes in the reinitialization process, is added to the assumptions previously listed, the total reinitialization time for 10,000 realizations is approximately 73 seconds. Here again the effect of adding an S element causes twice the increase in processing time as does the addition of G element.

7.5 The Simulation Program

The simulation program may be further divided into three sub-programs. Subroutines SCAC and SCNT will be treated in separate sections. The short control program shown in Figure 4.24 is the third portion and it will be discussed here. The initialization of TIME. CLKW, and the CLK register consumes 44.2 micro seconds plus 22.1 micro seconds for each additional word in CLK. The time for the cycling between SCAC and SCNT is 23.8 micro seconds plus 22.1 micro seconds for each additional word of CLK which must be examined. Note that the initialization takes place once each simulation, while the cycling takes place once each simulation plus once each transient in the system.

If an additional assumption that all elements undergo one and only one transient is added to the previous assumptions, then the time consumed in this portion of the simulation program is about 12 seconds. It was also assumed that at least one P signal was periodic so that the cycling routine required the examination of only a single word of CLK.

7.6 Subroutine SCNT

Some computational times of interest are shown in Tables VII-4 through VII-8. The search portion times are shown in the first of these tables. This portion is utilized on each system transient. Its contribution to total processing time can be estimated by adding to the previous assumptions the restriction that the transients occur singly and that each transient affects 1.2 other circuits in the system. This is not to be confused with fan-out since we are only interested in connections within the system being studied. It will also be useful later on to assume that the extra 0.2 interconnections are distributed in the same proportion as are the total system elements. In this case, for 10,000 realizations, the search portion of SCNT consumes 24 seconds for each transient or a total of 17.2 minutes. Again the S elements proved twice as slow to process as the G elements. Recall that

Table 7.4

Computational Times of Interest in Subroutine SCNT

(Scanning portion)

A. Entrance, Initialization, and Exit	30.6 μ sec
B. Processing a word of TRN with no 1's*	35.7 μ sec
C. Processing a word of TRN with at least one* 1	
1) processing the word itself	37.4 μ sec
2) processing matrix line with no active input	37.4 μ sec
3) processing matrix line with an active input (up to call to SCAT)	64.6 μ sec
D. Subroutine SCAT, regardless of nature of the matrix line element	47.9 μ sec

*Subtract 20.4 μ sec for the last word.

Table 7.5

Computational Times of Interest in Subroutine SCNT

(AND routines).

Let W_C be the number of computer words in CLK scanned in ERSE.

W_T be the number of computer words in CURR.

W'_T be the number of the word in CURR where the first discrepancy between MATR and CURR is found.

In these routines the times are not fixed so that an average processing time is given. In all cases the actual time will differ by no more than 6.8 μ sec.

A. Time to process an input transient which maintains the circuit in the ON state

$$46.8 + 39.1\ (W_T - 1)\ \mu\text{sec}$$

B. Time to process an input transient which makes the circuit to go from the OFF-GOINGS state to the ON state

$$170.8+20.4(W_C-1)+39.1(W_T-1)\mu\text{sec}$$

C. Time taken to process a signal which causes the circuit to go from the OFF state into the ON-GOING state

$$176.8+20.4(W_C-1)+39.1(W_T-1)\mu\text{sec}$$

D. Time taken to process a signal which causes the circuit to go from the ON state to the OFF-GOING state

$$203.8+20.4(W_C-1)+39.1(W'_T-1)\mu\text{sec}$$

E. Time taken to process a signal which causes the circuit to go from the ON-GOING state to the OFF state

$$186.0+20.4(W_C-1)+39.1(W'_T-1)\mu\text{sec}$$

F. Time taken to process a signal which maintains the circuit in the OFF state

$$46.8 + 39.1\ (W'_T - 1)\ \mu\text{sec}$$

Table 7.6

Computational Times of Interest in Subroutine SCNT

(OR-NOR routines)

Let W_C'' be the number of the first word in CURR where MATR and CURR have at least one bit in common.

W_C, W_T, W'_C be as defined in Table VII-5

Again average processing times are listed with actual times always lying within ± 6.8 μ sec

A. Time to process an input signal which maintains a circuit in the ON state

$$41.7 + 32.1\,(W_T''-1)\ \mu\text{ sec}$$

B. Time to process an input transient which causes the circuit to change from the OFF state to the ON-GOING state

$$188.4+20.4(W_C-1)+32.1(W_T''-1)\mu\text{sec}$$

C. Time to process an input transient which causes the circuit to go from the OFF-GOING state to the ON state

$$183.6+20.4(W_C-1)+32.1(W_T''-1)\mu\text{sec}$$

D. Time taken to process an input transient which causes the circuit to go from the ON-GOING state to the OFF state

$$169.9+20.4(W_C-1)+32.1(W_T-1)\mu\text{sec}$$

E. Time to process an input transient which causes the circuit to go from the ON state to the OFF-GOING state

$$158.3+20.4(W_C-1)+32.1(W_T-1)\mu\text{sec}$$

F. Time to process an input transient which causes the circuit to be maintained in the OFF state

$$41.7 + 32.1\,(W_T-1)\ \mu\text{ sec}$$

Table 7.7

Computational Times of Interest in Subroutine SCNT

(RST Subroutines)

A.	Time consumed to examine a wrong polarity transient	28.9 μ sec
B.	Time to process a transient on the R input if the device is already reset	57.1 μ sec
C.	Time to process a transient on the S input if the device is already set	58.8 μ sec
D.	Initial processing before search for first matrix line	
	1) R input	89.1 μ sec
	2) S input	90.4 μ sec
	3) T input	87.0 μ sec
E.	Determination of first line time	
	1) Input is in first matrix line	35.7 μ sec
	2) Input is in second matrix line	59.5 μ sec
	3) Input is in third matrix line	83.3 μ sec
F.	Change of state time, first line	
	1) Element previously in steady state	$127.5+20.4(W_C-1)\mu$ sec
	2) Element previously in transient state	$139.4+20.4(W_C-1)\mu$ sec
G.	Change of label time	
	1) Element contains one matrix line	56.1 μ sec
	2) Element contains two matrix lines	96.9 μ sec
	3) Element contains three matrix lines	137.7 μ sec
H.	Exit Time	13.6 μ sec

Table 7.8

Computational Times of Interest in Subroutine SCNT

(JK Subroutines)

A. Time to exit if status, polarity of transient, are wrong for particular input

1)	Positive transient on J, K or G input	52.7 μ sec
2)	Positive transient on PJ, status =1	64.6 μ sec
3)	Positive transient on PK, status =0	62.9 μ sec
4)	Negative transient on J, status = 0	54.4 μ sec
5)	Negative transient on K, status =1	62.9 μ sec
6)	Negative transient on PJ, status = 1	66.3 μ sec
7)	Negative transient on PK, status = 0	71.4 μ sec

B. Initial processing before search for first matrix line

1)	Positive transient, PJ input	102.0 μ sec
2)	Positive transient, PK input	98.6 μ sec
3)	Negative transient, J input	91.8 μ sec
4)	Negative transient, K input	100.3 μ sec
5)	Negative transient, PJ input	107.1 μ sec
6)	Negative transient, PK input	112.1 μ sec
7)	Negative transient, C input	113.8 μ sec

C. Time required to determine first matrix line

1)	First line active	40.8 μ sec
2)	Second line active	66.3 μ sec
3)	Third line active	97.8 μ sec
4)	Fourth line active	116.3 μ sec
5)	Fifth line active	141.8 μ sec

Table VII-8 continued - - - - -

--- --- Table 7.8 continued

D. Time required to build the input status word

1) Unconnected input	27.2 μ sec
2) Connected input 1 entry	71.4 + 27.2(W-1)μ sec
0 entry	81.6 + 27.2(W-1)μ sec

(Where W is the number of the interconnection word containing the input.)

E. Comparison of input status word

1) Check on first pass	30.6 μ sec
2) Check on second pass	59.5 μ sec
3) Failure to match	51.0 μ sec

F. Change of state time - Same as Table 7.7F.

G. Change of label time

1) Element contains one matrix line	56.1 μ sec
2) Element contains two matrix lines	96.9 μ sec
3) Element contains three matrix lines	137.7 μ sec
4) Element contains four matrix lines	178.6 μ sec
5) Element contains five matrix lines	219.5 μ sec

H. Exit time 13.6 μ sec

this computation did not include the element processing.

Table 7.5 shows some computational times for the AND subroutines. It should be noted that routine also serves the NAND elements with no difference in processing times. Assuming that half of the G elements in the system are AND or NAND elements and that the state changes are equally divided between ON-GOING and OFF-GOING, the AND-NAND subroutines contribute an additional 40 seconds.

Table 7.6 shows some computational times involved in the OR-NOR subroutines. If it is required that the OR-NOR circuits experience equally ON-GOING and OFF-GOING state changes, then the contribution of the OR-NOR subroutines to the total processing time is about 35 seconds.

Computational times pertinent to the RST element processor are shown in Table 7.7 . It will be noted that the time to process an RST element depends on which input is activated, and how many matrix lines are used to specify the element. An order of magnitude may be obtained by assuming 8 of the S elements have only the T input connected, another 8 have the R and S inputs connected with triggering equally likely between them, while 4

have all the inputs connected with the transient active on the T input. It will also be assumed that all S elements are in the steady state when an input transient appears. The total time consumed is approximately 90 seconds.

It is worth noting that approximately 25% of the processing time for the AND-NAND, OR-NOR, and RST elements was involved in loading the CLK register.

Pertinent timing information for the JK processor is contained in Table 7.8 . If a relatively crude estimate of the difference between RST and JK processing times can be obtained by assuming that the twenty flip-flops were JK elements, with 15 using only three inputs while 5 use five inputs. This would result in an increase over RST processing time of approximately 96 seconds .

7.7 Subroutine SCAC

The processing times for subroutine SCAC are extremely varied depending on circumstance, so that any estimate here will involve greater approximations than those made heretofore. Some processing times of interest are indicated in Table 7.9 . A very crude estimate of the processing time for the system previously

Table 7.9

Computational Times of Interest for Subroutine SCAC

A.	Initialization of Routine	34.0 μ sec
B.	Exit to Line Processor	
	1) If $A_1 = 1$	18.7 μ sec
	2) Time for normalizer (each bit) N_C is number of shifts required	$37.7+0.34N_C \mu$ sec
	3) Exit from normalize loop	10.5 μ sec
	4) Change word in CLK	28.9 μ sec
C.	Exit from SCAC including modification of CURR	$57.8+35.7(W_T-1)\mu$ sec
D.	Time to modify clock increment if the clock has not gone to zero (ave)	55.4 μ sec
E.	If clock has gone to zero, the time to determine the nature of element in the line	
	1) Gate element	40.8 μ sec
	2) P element	47.6 μ sec
	3) S element	45.9 μ sec
F.	Time to process a G line	$185.3+40.8(W_C-1)\mu$ sec
G.	Time to process an S element where L is the number of matrix lines required for the S element	$408.4 + 81.6\ (W_C-1)+37.4(L-1)\mu$sec
H.	Time to process a P line	
	1) if the entire clock word is not zero	23.8 μ sec
	2) if a level changes	$198.9+40.8(W_C-1)\mu$ sec
	3) if single pulse, leading edge	$185.3+20.4(W_C-1)\mu$ sec
	4) if single pulse, trailing edge	$198.9+40.8(W_C-1)\mu$ sec
	5) periodic pulse, (average)	$187.0+20.4(W_C-1)\mu$ sec

used as an example yield a total time of nine and three quarter minutes. Additional assumptions were. The contents of CLK were presumed to contain 10 ones on the average, but only one of these represented a transient, the others resulted in comparison with the clock decrement word.

It should be stressed, that with the assumptions made, the results are only crude estimates. They do, however, allow order of magnitude estimates.

7.8 Processing Time of the Example.

The total time for processing the circuit of the example through 10, 000 simulations would be about 30 minutes or about 180 micro seconds per simulation. About one half the total time was used in the processing of subroutine SCNT. Of this time about 5/8 was consumed evaluating words in the input matrix which did not contain inputs with transients on them.

One is tempted to consider the preparation of a routine which forms an output matrix from the information in the input matrix, and to use these matrix lines during SCNT in an identical manner to the way CLK is used during SCAT. However, careful consideration shows that no saving in time is obtained if each

element undergoes but a single transient in the simulation, and there are only a small number of elements in the system under consideration. The idea would have merit, however, in those cases where a large system is to be simulated, or when a small system is to be simulated over several cycles of operation, i.e., the entire counting sequence of a counter. If such tasks are to be encountered to any large degree, it is the opinion of this writer that the use of an output matrix be thoroughly investigated.

CHAPTER VIII

RECOMMENDATIONS FOR FUTURE WORK

There are two major areas in which more work could be profitably undertaken; the expansion of the simulator, and a study of the effects of packaging correlation on element statistics.

The simulator as shown in this thesis could be suitably modified to serve as a synthesizer. The main program could be used, in conjunction with some additional routines and possibly some added hardware, to produce a display on an oscilloscope of the system wave shapes. In this case nominal delays would be used for the elements. The major problems would be interfacing the computer output with the oscilloscope, and "smoothing" the unequal time increments which occur between transients.

The effects of element packaging on the statistics are not well known at present, but they are becoming increasingly more important. The trend in packaging today appears to be toward more circuits in the same envelope. In the monolithic circuits discussed here, this means that the circuits in one envelope would have been fabricated from the same silicon wafer and manufactured under identical environments. It is reasonable

that the assumption of statistical independence used herein would be much less valid for elements in the same envelope. A suitable field for further investigation would be the nature of the correlation between elements in the same envelope and the means for introducing the effects of the correlation into the computer program.

Appendix A. Convolution by Means of Impulses

It is a well known theorem from probability (see, for example, Davenport and Root[13], page 36) that the probability density function of the sum of two statistically independent random variables is obtained by convolving the two input density functions. That is, if

$$Z(t) = X(t) + Y(t) \tag{A-1}$$

then

$$p_Z(t) = \int_{-\infty}^{\infty} p_x(T)\, p_y(t-T)\, dT. \tag{A-2}$$

If the functions can be represented by curves which fulfill the usual conditions, the Laplace transform of equation A-2 may be taken, yielding

$$P_z(S) = P_x(S)\ P_y(S) \tag{A-3}$$

If now both sides of equation A-3 are multiplied by S^2 it obtains

$$S^2 P_z(S) = S\ P_x(S)\quad S\ P_y(S) \tag{A-4}$$

If there exists a time t_0 such that for all $t < t_0$, $p_z(t)$, $p_y(t)$ and $p_x(t)$ along with $p_z'(t)$ are all equal to 0, then the inverse Laplace transform of equation A-4 may be taken yielding:

$$p_z''(t) = \int_{-\infty}^{\infty} p_x'(T) \; p_y'(t-T) \, dT \qquad \text{(A-5)}$$

Repeated applications of the above principles can yield the following useful result:

If there exists a value of time t_0 such that for all $t < t_0$; $p_x(t)$, $p_y(t)$, $p_z(t)$ and all the higher order derivatives are zero, then;

$$p_z^{j+k}(t) = \int_{-\infty}^{\infty} p_x^j(T) \; p_y^k(t-T) \, dT \qquad \text{(A-6)}$$

where the superscripts refer to the order of the derivative.

If the input density functions can be at least piecewise approximated by an Nth order power function, it is only necessary to differentiate it N + 1 times to produce a train of weighted impulses. The convolution of the impulse trains is merely a matter of adding the ordinates and multiplying the weights. The resultant pulse train is then the Mth derivative of the sum density function

and can be easily evaluated by integration. The process is perhaps best illustrated by a simple numerical example. Consider in Figure A-1 the two density functions for the variables x(t) and y(t), possibly representing propagation delays through cascaded logic elements. The derivatives can be expressed as follows:

$$p'(x) = \sum_{j=0}^{1} a_j \, I\,(t-t_j) \qquad \begin{aligned} a_0 &= \frac{1}{100} & t_0 &= 300 \\ a_1 &= \frac{-1}{100} & t_1 &= 400 \end{aligned} \tag{A-7}$$

$$p'(y) = \sum_{k=0}^{1} b_k \, I\,(t-t_k) \qquad \begin{aligned} b_0 &= \frac{1}{250} & t_0 &= 250 \\ b_1 &= \frac{-1}{250} & t_1 &= 500 \end{aligned} \tag{A-8}$$

Convolution is obtained by taking each impulse of p'(x) and multiplying its strength by each impulse of p'(y) in turn and placing it at an ordinate which is the sum of the ordinates of the impulses involved

$$p''(z) = \sum_{j,\,k\,=\,0,\,0}^{1,\,1} c_{jk} \; I \; [\, t-(t_j + t_k)\,] \tag{A-9}$$

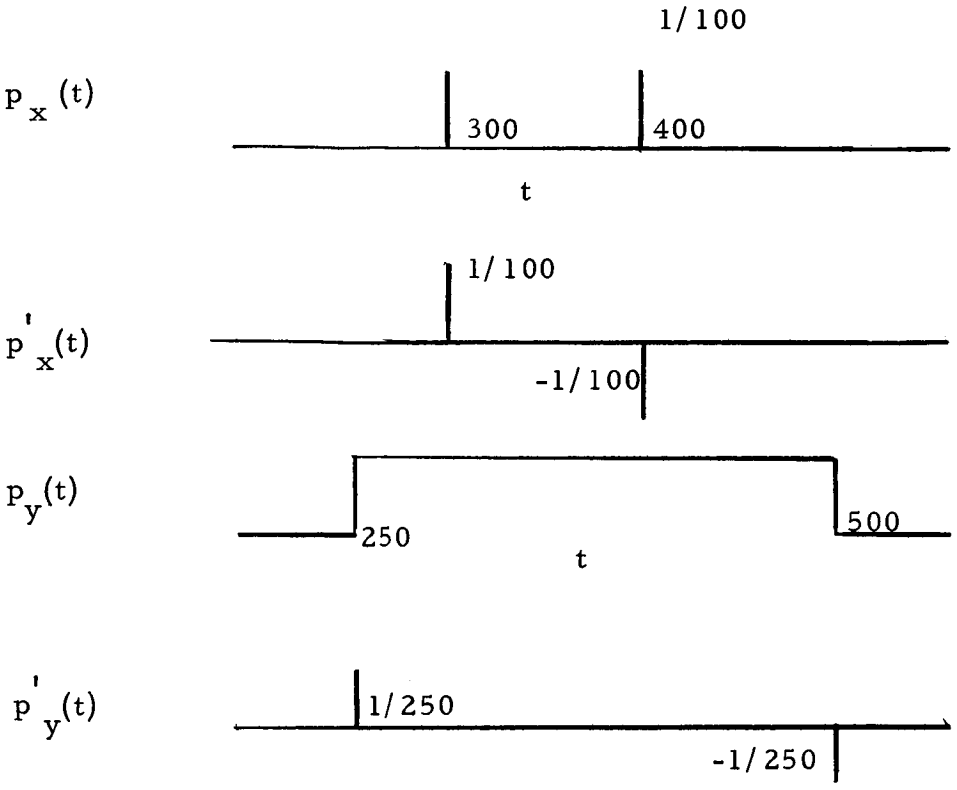


Figure A-1 (a) Input Density Functions and Their Derivatives

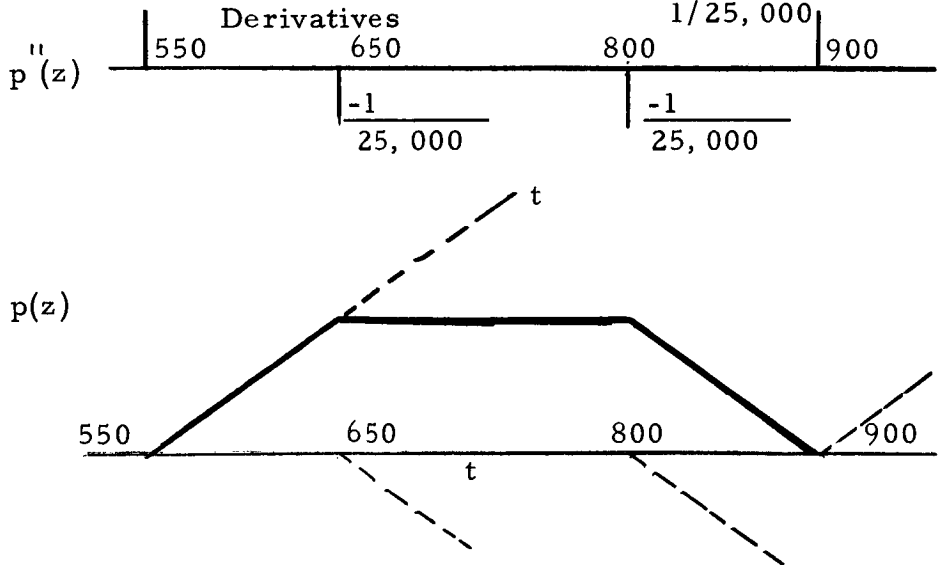


Figure A-1 (b) Output Density Function Derivatives and Integration

Figure A-1 Impulse Convolution Example

where

jk	c_{jk}	$t_j + t_k$
00	$\frac{1}{25,000}$	550
01	$\frac{-1}{25,000}$	800
10	$\frac{-1}{25,000}$	650
11	$\frac{1}{25,000}$	900

Since the desired result is usually p(z) it is necessary to doubley integrate equation A-9 which yields

$$p(z) = \sum_{j,k=0,0}^{1,1} c_{jk} [\, t-(t_j+t_k) \,] \; U[\, t - (t_j+t_k) \,] \qquad \text{(A-10)}$$

$$U(x) = \text{unit step}$$

which is shown also in Figure A-1.

This result can easily be generalized to where the $l^{\underline{th}}$ derivative of p(x) was required along with the $m^{\underline{th}}$ derivative of p(y), in that case

$$p(z) = \frac{1}{(l+k-1)!} \sum c_{jk} [\, t - (t_j+t_k) \,]^{l+k-1} \; U[\, t - (t_j+t_k)] \qquad \text{(A-11)}$$

Any reader with further interest in this technique is referred to Guillemin[14] for an excellent treatment especially on the accuracy of curve fitting, and to Gray[4] for an interesting design application.

The arguments in this appendix are based on the fact that the probability density functions in question can be satisfactorily represented by curves which are Laplace transformable. In most cases of interest this is true. However, a more rigorous derivation is contained in Appendix II of Gray's[4] book.

Appendix B. A Derivation

It is desired to show that:

$$\frac{M}{2}\left[\mu_+^2 + \mu_-^2\right] \leq \sum_{j=1}^{M} \mu_j^2 \qquad \text{(B-1)}$$

Since the case M = 2 is trivial ($\mu_1 = \mu_-$, $\mu_2 = \mu_+$) it will be assumed that M > 2 and also that M is even. If M is odd and the last element in the chain is unsorted, the result is merely the addition of a term μ to each side of equation B-1. It may be helpful to refer to Figure B.1 in what follows.

Consider, first, the lower sort, it will be shown that:

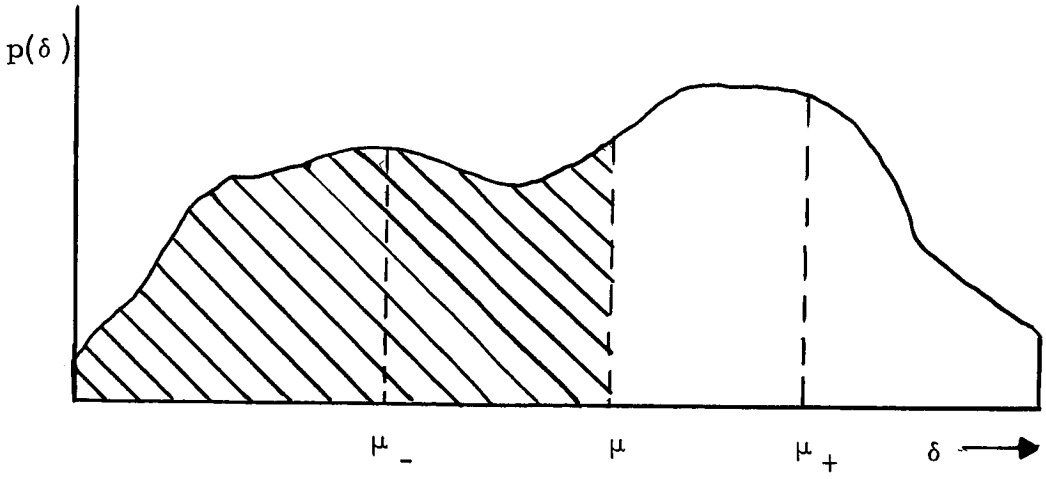
$$\frac{M}{2}\mu_-^2 \leq \sum_{j=1}^{M/2} \mu_j^2 \qquad \text{(B-2)}$$

Note that from equation 2.6,

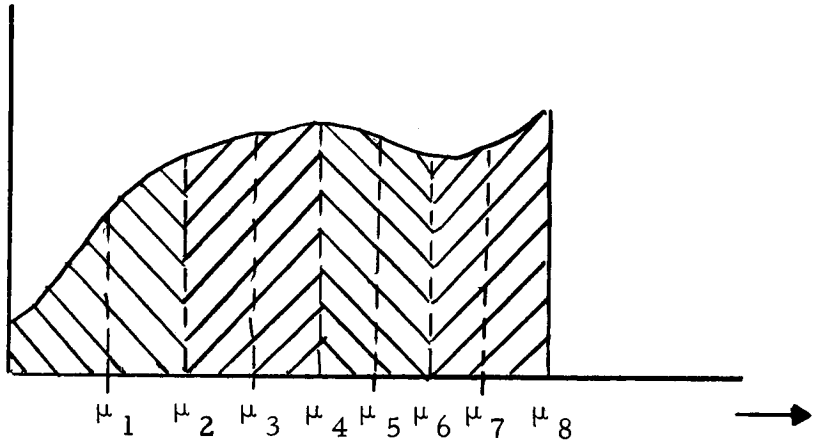
$$\frac{M}{2}\mu_- = \mu_1 + \mu_2 + \ldots + \mu_{M/2} \qquad \text{(B-3)}$$

Squaring yields

$$(\mu_-)^2 = \left(\frac{2}{M}\right)^2 (\mu_1 + \mu_2 \ldots \mu_{M/2})^2 \qquad \text{(B-4)}$$



(a) A Sort by Two's



(b) A Possible Sort with M Even, Lower Half Shown Only.

Figure B.1 Reference Sketch for the Derivation

Substituting into B-2, and rearranging, yields

$$0 \leq \frac{M}{2} \sum_{y=1}^{M/2} \mu_j^2 - (\mu_1 + \cdots \mu_{N/2})^2 \qquad \text{(B-5)}$$

or equivalently

$$0 \leq (\frac{M}{2} - 1) \sum_{j=1}^{M/2} \mu_j^2 - 2 \sum_{j=1, k<j}^{M/2} \mu_j \mu_k \qquad \text{(B-6)}$$

Consider the function

$$0 \leq \sum_{j=1}^{M/2 - 1} \sum_{k \; j} (\mu_j - \mu_k)^2 \qquad \text{(B-7)}$$

This will have each μ_j^2 appearing $(\frac{M}{2} - 1)$ times and each cross product term appears once with a negative sign. Thus B-7 is equivalent to B-6 and equation B-2 is proven.

A similar method shows that

$$\frac{M}{2} \mu_+^2 \leq \sum_{j=M/2}^{M} \mu_j^2 \qquad \text{(B-8)}$$

adding equations B-8 and B-2 yields

$$\frac{M}{2} [\, \mu_+^2 + \mu_-^2 \,] \leq \sum_{j=1}^{M} \mu_j^2 \qquad \text{(B-9)}$$

Appendix C. The Generation of Random Delays

C.1 A Random Number Generator for the DDP-116

There is a well known[15, 16] algorithm for the generation of a string of pseudo-random numbers given by

$$R_j = R_{j-1} \times 5^K \pmod{2^N} \qquad (C.1)$$

where R_j = j^{th} random number in the string

N = number of binary positions in R_j

K = largest odd number such that $5^K \leq 2^N - 1$

R_0 is chosen to be any odd number.

The sequence of random numbers so generated is periodic with period 2^{N-2} and will be uniformly distributed.

In the DDP-116, if single precision is used to generate the random numbers, the period will be 2^{13} (or 8192); double precision on the other hand yields a period of 2^{28} (or 268, 435, 456). For applications of any appreciable size, the double precision seems desirable, so that a program for double precision was preferred.

Since $5^{11} < 2^{30} < 5^{13}$,

a value of K = 11 was chosen. If M denotes 5^{11} then M in octal is given by

$$M = 03022, 07335 \qquad (C.2)$$

To use the double precision in the DDP-116, each number is considered to be the sum of two 15 digit numbers as is shown in the following example:

$$M = 03022\ 00000 + 00000\ 07335$$

or more briefly,

$$M = M_1 \phi + \phi M_2 \qquad (C.3)$$

The ϕ in equation (C.3) represents a block of 15 binary zeroes. M_1 and M_2 represent the 15 most significant digits (MSD) and 15 least significant digits (LSD) respectively of M. Similarly

$$R_j = R_{j1} \phi + \phi R_{j2}. \qquad (C.4)$$

The sixty bit result of multiplying M and R_j is given by:

$$R_{j+1} = [M_1 R_{j1}]\ \phi\phi + \phi[M_1 R_{j2}]\ \phi + \phi\ [M_2 R_{j1}]\ \phi + \phi\phi\ [M_2 R_{j2}] \qquad (C.5)$$

where the brackets represent the results of the normal machine

multiplication of 15 digit words. Since the algorithm requires multiplication mod 2^{30} the 30 most significant places are thrown away and we can write the new random number as

$$R_{j+1} = R_{(j+1)1}\,\phi + \phi\,R_{(j+1)2} \qquad (C.6)$$

where $R_{(j+1)2}$ = the LSD of $[M_2 R_{j2}]$

$R_{(j+1)1}$ = the LSD of $[M_1 R_{j1}]$ plus the LSD of $[M_2 R_{j1})$ plus the MSD of $[M_2 R_{j2}]$

The double precision generation, it should be stressed, was chosen to obtain a large period for the pseudo-random string of numbers. For the applications of this thesis 2^{15} (or 32, 768) different numbers were considered sufficient, so that only R_{j1} was used in the Monte Carlo analysis itself. R_{j1} was selected over R_{j2} because the numbers become more random toward the middle.

The subroutine developed is given below. It is assumed that R_0, the initial value, is present in memory locations RA and RB for R_{j1} and R_{j2} respectively. It is sometimes desirable if, for example, an experiment is to be repeated, to begin at an arbitrary

starting point. If more randomness of starting point is desired the real time clock could be made negative and used as an initial value.

LABEL	OPERATION	ADDRESS	COMMENT
RGEN	DAC	**	Store Exit Address
	LDA	= '03022	Load M1 into accumulator
	MPY	RB	Form [M_1 R_{j2}]
	IAB		Pick LSD
	STA	TMP1	Store in TMP1
	LDA	RA	Load RA into accumulator
	MPY	= '07325	Form [R_{j1} M_2]
	IAB		Pick LSD
	ADD	TMP1	Add to previous result
	SSP		Ignore overflow
	STA	TMP1	Store in TMP1
	LDA	= '07325	M_2
	MPY	RB	Pick MSD of [R_{j2} M_2]
	ADD	TMP1	Add to previous result
	SSP		Ignore overflow
	STA	RA	$R_{(j+1)1}$
	IAB		Pick LSD of [R_{j2} M_2]

STA	RB	$R_{(j+1)2}$
LDA	RA	$R_{(j+1)1}$ into accumulator
JMP*	RGEN	EXIT

The subroutine shown was implemented and tested. In figure C.1, the results of a run of 25,000 numbers are shown. The "goodness of fit" (Chi squared test) on runs of this type was approximately 90%.

Note that the round off process used in tabulation has made a trapezoidal effect at the edges. The work uniform in this thesis will generally refer also to these trapezoidal density functions.

C.2 Distributions other than Uniform

The random number generator previously described generates numbers to a uniform distribution. By suitable manipulation, it can also be used to produce random numbers fitted to other density functions. Use is made of the fact that, if the random variable y is a monotonic function of the random variable x then;

$$p(y) = p(x) \frac{dx}{dy} \qquad (C.7)$$

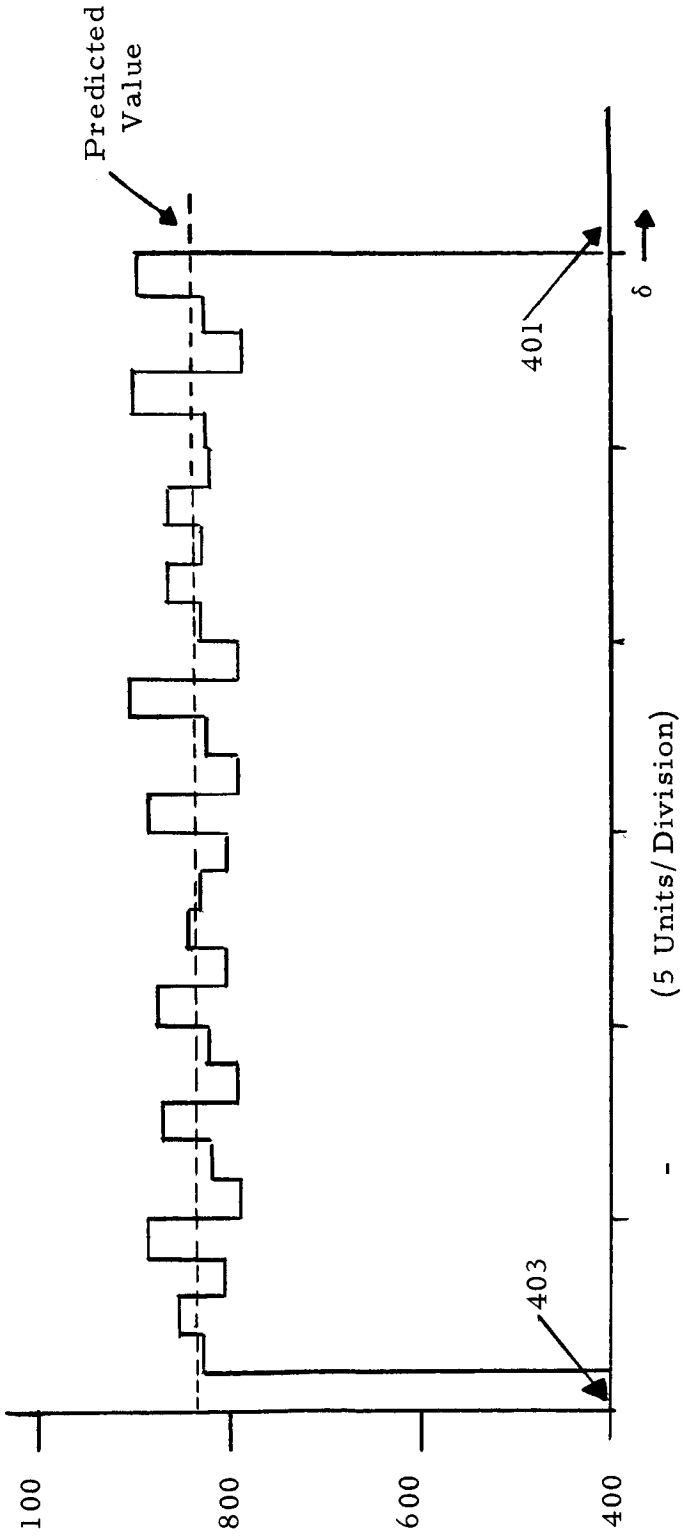


Figure C.1 Test Run of Uniform Density Generator (25, 000 Samples)

Considering p(x) to be the uniform density generated by the pseudo-random number generator and p(y) to be the desired density function, the relation can be integrated and a relation between y and x evolved. The technique is best illustrated by an example.

Suppose it is desired to produce numbers having a sloped distribution as is shown in Figure C.2.

Here

$$p(x) = 1 \qquad 0 < x < 1$$

$$= 0 \qquad \text{elsewhere}$$

$$p(y) = p(a) + m(y-a)$$

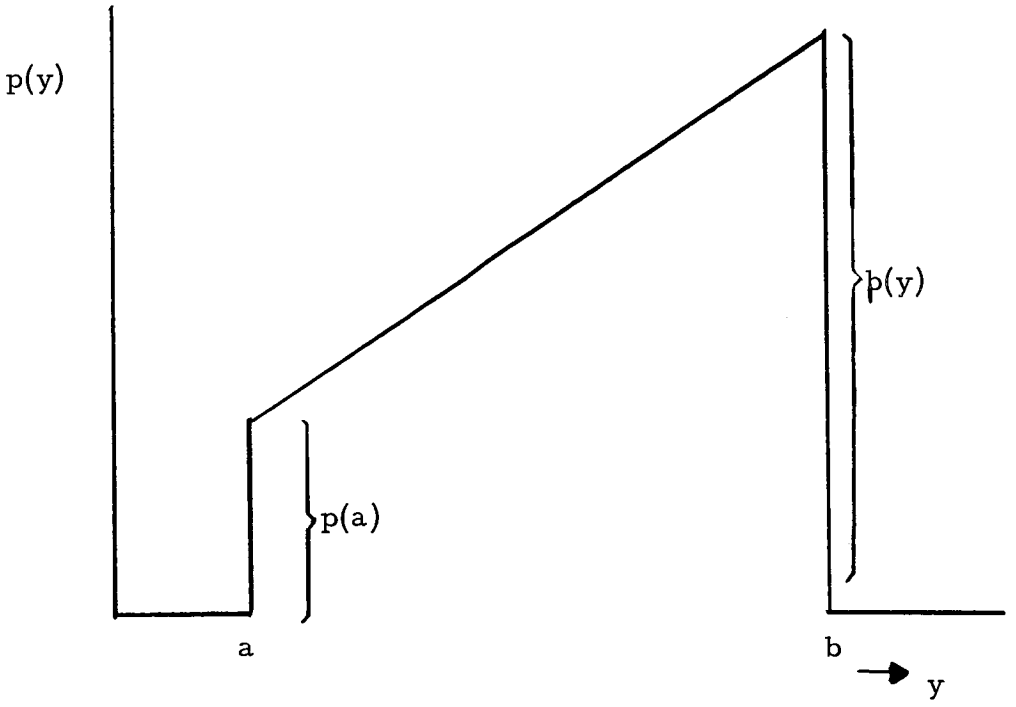
where m is the slope of the line.

Using equation (C.7) it follows that

$$[\, p(a) + m(y-a) \,]\, dy = dx \tag{C.8}$$

Upon integration

$$p(a)\,(y-a) + \frac{m}{2}(y-a)^2 = x + c \tag{C.9}$$



$$\text{MBAR} = \frac{b - a}{p(b) - b(a)}$$

Figure C.2 Sloped Density Function Parameters

If we impose the condition that at $x = 0$, $y = a$ then $c = 0$.

A second check is obtained by requiring that at $x = 1$, $y = b$, then

$$c = p(a)\,(b - a) + \frac{m}{2}\,(b - a)^2 - 1 \qquad (C.10)$$

Note that $p(a) = \frac{1}{b-a} - m\,\frac{(b-a)}{2}$

then $$c = 1 - m\,\frac{(b-a)^2}{2} + \frac{m}{2}(b-a)^2 - 1 = 0$$

The required relationship is then:

$$(y-a)^2 + \frac{2}{m}\,(y - a)\,p(a) - \frac{2}{m}\,x = 0 \qquad (C.11)$$

which when solved for y-a yields

$$y-a = \frac{-p(a)}{m} \pm \sqrt{\frac{p(a)}{m} + \frac{2x}{m}} \qquad (C.12)$$

Since we require that y=a for x=0, the positive sign is selected.

Thus, if m, a, and p(a) are given each value of x produced by the pseudo random number generator can be used to produce a value of y. This procedure was implemented on the DDP-116 using the program shown below. Most of the complexity

occurs because the routine is double precision, treating the integer and fraction portion of the number separately. Also the available square root subroutine assumes that the number to be rooted is a left or left minus 1 justified fraction. Assume that x(X), $\frac{1}{m}$(MBAR), a(A), and p(a) (PA) are present in memory

```
        CRA
        STA     SHCT        Initialize shift counter to zero
        LDA     PA
        MPY     MBAR        Form p(a)/m
        STA     INT1        Load MSD in INT1
        SMI                 Check sign
        JMP     PLS1        If sign positive go to PLS1
        IAB                 If sign negative make sign
        SSM                 of LSD negative too
RET1    STA     FRC1        Load LSD into FRC1
        LDA     X
        MPY     MBAR        Form x/m
        LLS     1           double. Now 2x/m
        STA     INT2        Load MSD into INT2
        SMI                 Check sign MSD
```

	JMP	PLS2	If positive go to PLS2
	IAB		If negative make sign of LSD
	SSM		negative too
RET2	STA	FRC2	Store LSD in FRC2
	LDA	INT1	
	MPY	INT1	Form $\left(\frac{p(a)}{m}\right)^2$ MSD x MSD
	IAB		
	ADD	INT2	Add to MSD of 2x/m
	STA	INT2	Store in INT2
	LDA	INT1	Form $\left(\frac{p(a)}{m}\right)$ LSD x MSD
	MPY	FRC1	
	LLS	1	Double
	ADD	INT2	Add MSD of result to INT2
	STA	INT2	
	IAB		
	ADD	FRC2	Add LSD of result to FRC2
	SSC		Check for overflow
	JMP	GO	If no overflow, jump to GO
	IRS	INT2	If overflow, Add 1 to INT2
	SSP		Set sign of LSD Positive
GO	STA	FRC2	
	LDA	FRC1	

```
        MPY     FRC1    Form (p(a)/m)^2 LSD x LSD
        ADD     FRC2    Add MSD of result into FRC2
        SSC             Check overflow
        JMP     LOAD    If no overflow
        IRS     INT2    If overflow add 1 to INT2
        SSP             Set sign FRC2 positive
LOAD    STA     FRC2
        IAB
        LDA     INT2    Set up double precision number
        SZE             for square rooting
        JMP     SHIF    In integer non-zero go to SHIF
PREP    LDA     SHCT
        TCA
        ANA     ='77    Number of de-normalizing shifts
        STA     TEMP    Extract number of shifts after
        LDA     VSHF    rooting
        ANA     ='177700 Extract order type
        ERA     TEMP    Create variable shift instruction
        STA     VSHF
        IAB             Load accumulator
        ANA     ='177776 Extract out pathological case
        CALL    SQRX2
```

```
        IAB                 Treat as fraction
VSHF    LLS     **          De-normalize
        SUB     INT1        Form y-a
        STA     INT         MSD into INT
        LDA     FRC1
        SMI                 Test for negative
        JMP     PLPL        Difference of two positive num-
                            bers.
        IAB                 If negative load with fractional
                            part of root
        SUB     FRC1        Positive number minus negative
        SSC                 Check for overflow
        JMP     EXIT        If no overflow
        IRS     INT         If overflow increment INT
        SSP                 Change sign
        JMP     EXIT
PLPL    IAB                 Load with fractional part of root
        SUB     FRC1        Test against FRC1
        STA     FRC
        SMI                 Test for negative
        JMP     *+6         If positive
        TCA                 Form 1 - fraction
        STA     FRC
        LDA     INT
```

```
        SUB     =1          Decrement INT
        STA     INT
        LDA     FRC
EXIT    SUB     ='40000     Round off
        SPL                 Round off
        RETURN              Result is in INT
        IRS     INT         Add 2 to INT
        RETURN              Result is in INT
PLS1    IAB
        JMP     RET1
PLS2    IAB
        JMP     RET2
SHIF    LRS     2           Normalize - number to be
                            rooted to a fraction, saving
        IRS     SHCT        number of double shifts re-
                            quired.
        SZE
        JMP     SHIF
        JMP     PREP
```

As it stands the integral portion of the number y is stored in INT, if full double precision is needed the round-off could be skipped and the number left in INT and FRC.

The results of generation of 25, 000 numbers with $p(a) = 0$, $m = 1/2048$, and $a = 10$ is shown in Figure C.3.

Figure C.4 shows the result of generating a triangular density function by adding successive samples from a uniform distribution of one half the width. Since the uniform density is actually trapezoidal due to round-off, it is expected that there will be rounding at the peak and at the extreme points.

C.3 An Example of Serial Correlation

It is axiomatic that man learns from his mistakes. What follows is the result of an error which occurred in the work of this thesis and it illustrates some of the problems associated with the use of random number generating algorithms.

During the preparation of the random number generator previously discussed a mechanical mistake in the computation of the highest odd power of 5 was made, and for a time, instead of 0302207335 (octal) the octal number 1454372001 was used as a multiplier. When it was tested for uniformity, it was approximately as good as the results shown in Figures C 3 and C.2. However, when it was desired to test the result of convolving three uniform density functions (or for that matter any multiple of three) the data

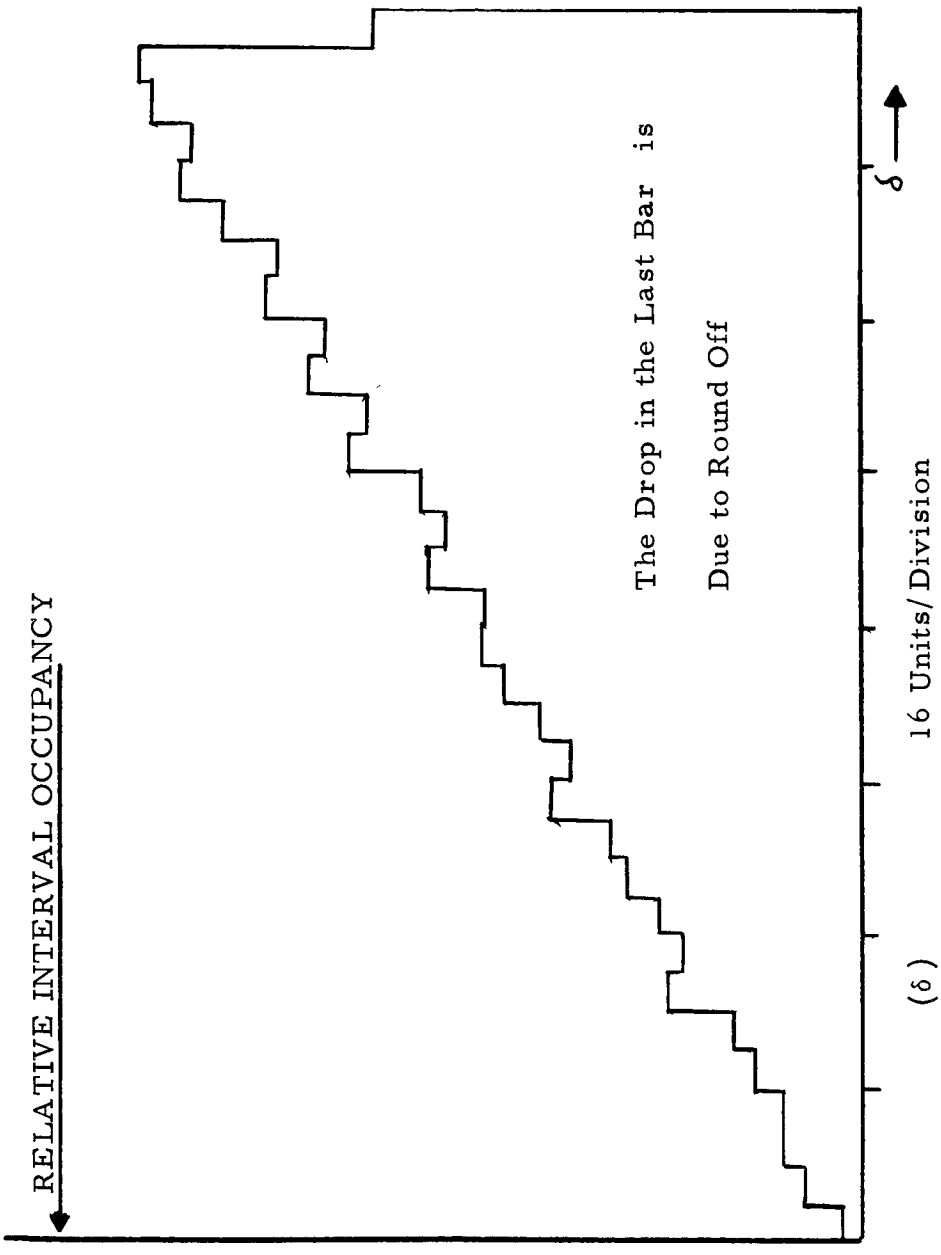


Figure C.3 Test Run (25,000 Samples) of Linear Density Generator

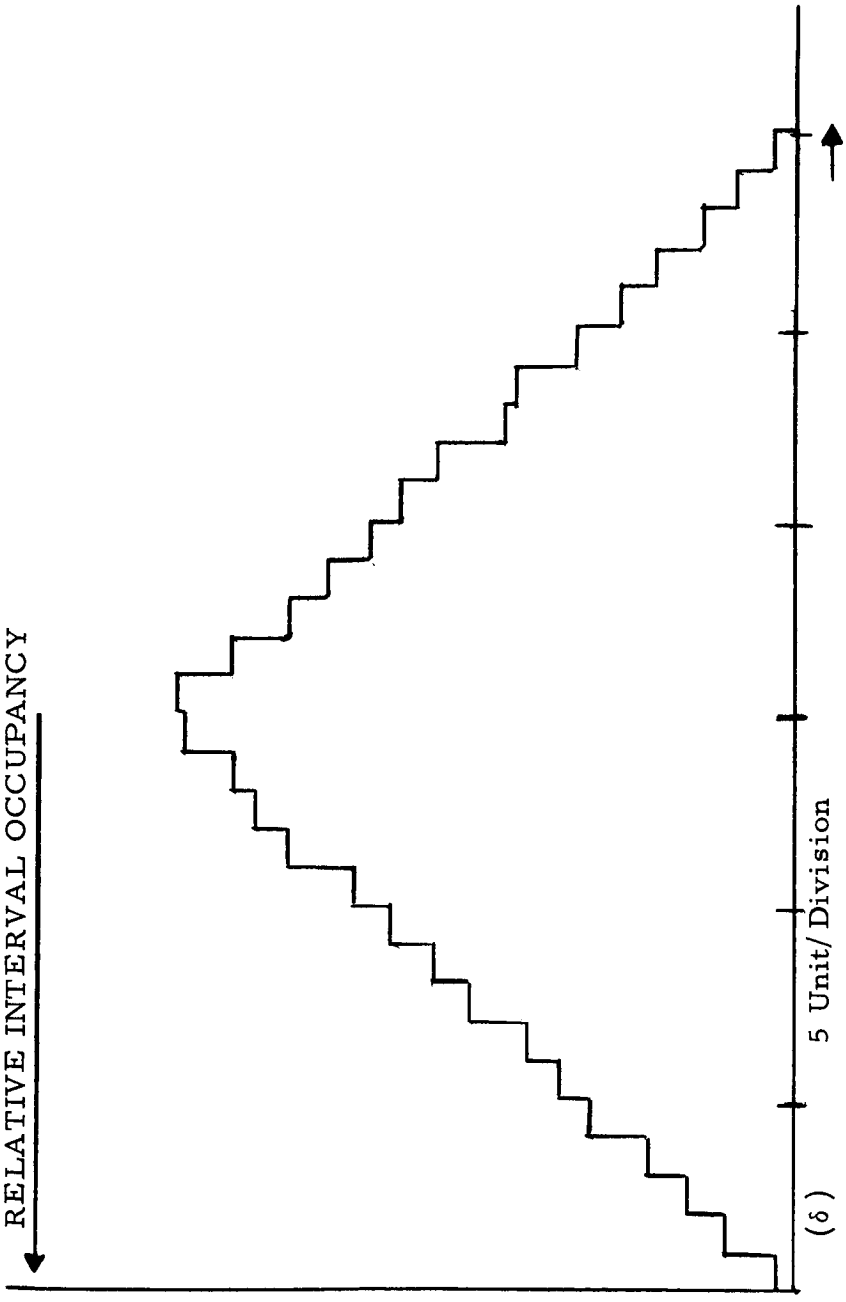


Figure C.4 Test Run of Triangular Density Generator (25, 000 Samples)

in Figure C.5 was obtained.

Note the "periodicity" of the results. It is as if there were a use, then a decay for two units. This effect is due to serial correlation and it is an effect which must be guarded against in Monte Carlo techniques. Basically, when three successive numbers generated by the algorithm are added, certain sums mod 6 are preferred over others. The user should be on the alert for phenomena of this nature. Steps outlined in Chapter III may be required.

There will be an excellent treatment of random number generating algorithms and their pathology in a forthcoming book by Knuth.[17]

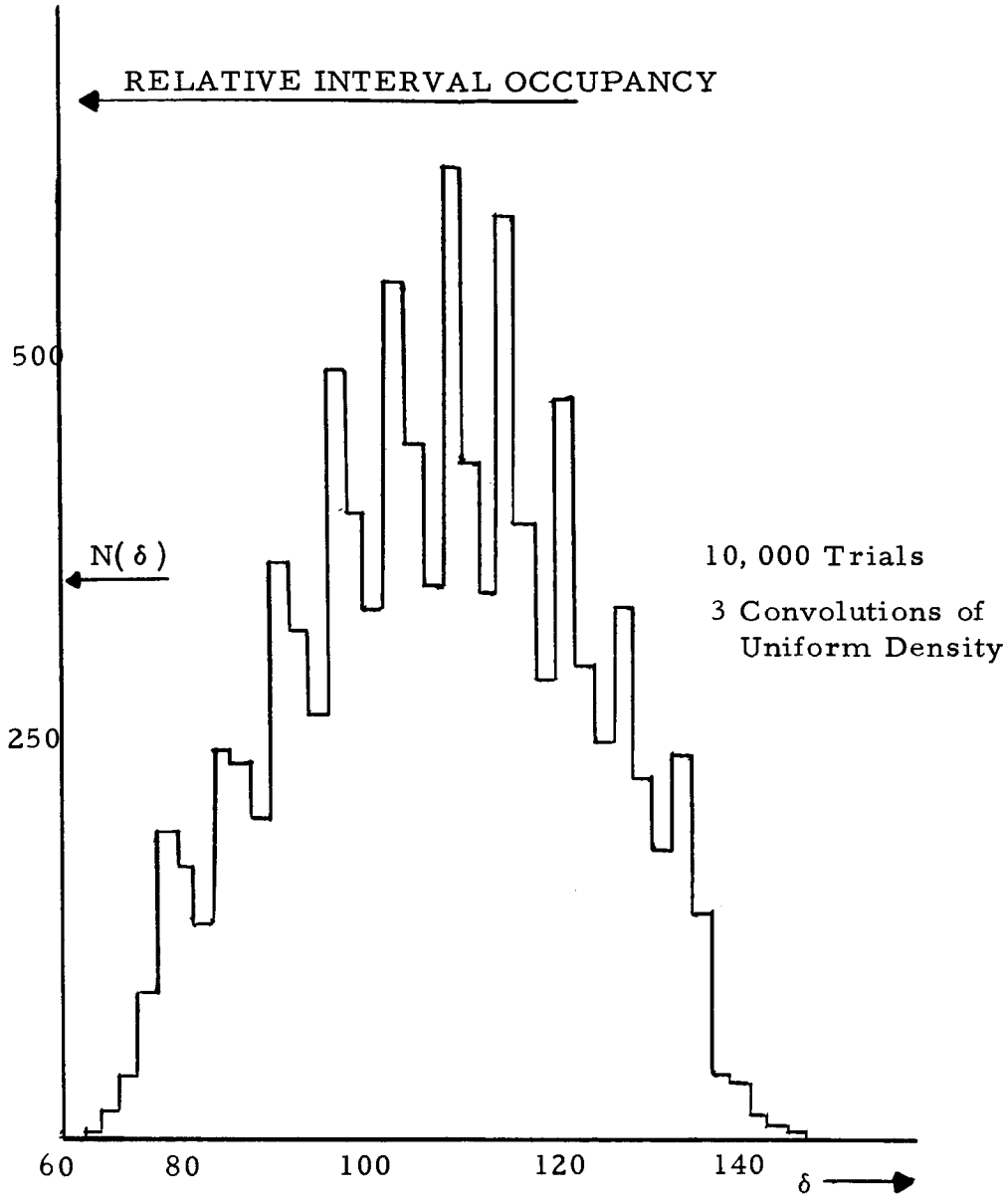


Figure C.5 An Example of Serial Correlation

Appendix D. The Applicability of the Simulator to Other Elements

D.1 General

It is noted that many digital systems will contain other types of elements than those discussed in the main body of the thesis. It is felt that most of the customarily utilized digital elements can be produced by combining element processors of the type discussed in detail. Several examples are listed below:

D.2 RST Flip-Flop with Steering Gate Inputs

This can be handled by redefining the RST element to have five inputs:

1) A Reset Input,

2) Reset Level input,

3) Set Input,

4) Set Level Input,

5) Toggle Input.

If the entries on card input list are arranged in this order, the main program modifications are relatively simple. Now, no distinction is required between RS and JK elements on the basis of inputs and the S element processor would have INCT

always set to five.

The new flow chart for the RST element is shown in Figure D.1. The reader will note that the effect of the changes relative to Figures 4.30 and 4.31 are:

(1) A transient on any level input is ignored.

(2) A positive transient on the set input requires, in addition to the earlier requirement that the circuit be reset, that the set level input also be 1.

(3) Similarly, a positive transient on the reset input now also requires a 1 on the reset level input.

D.3 Delay Multivibrator Element (DMV)

The delay multivibrator can be considered a combination of a one-input gate and an external level. The gate corresponding to the input could be treated as any other gate in the matrix, except that when its input is stimulated by a transient of the proper polarity it does not go into a transient state. Instead of a clock word, the corresponding word in the matrix line for the gate would indicate which of the P lines was to be triggered. The P line would be the same as that of a single pulse, the initial delay would correspond to propagation delay in the multivibrator and D1 would

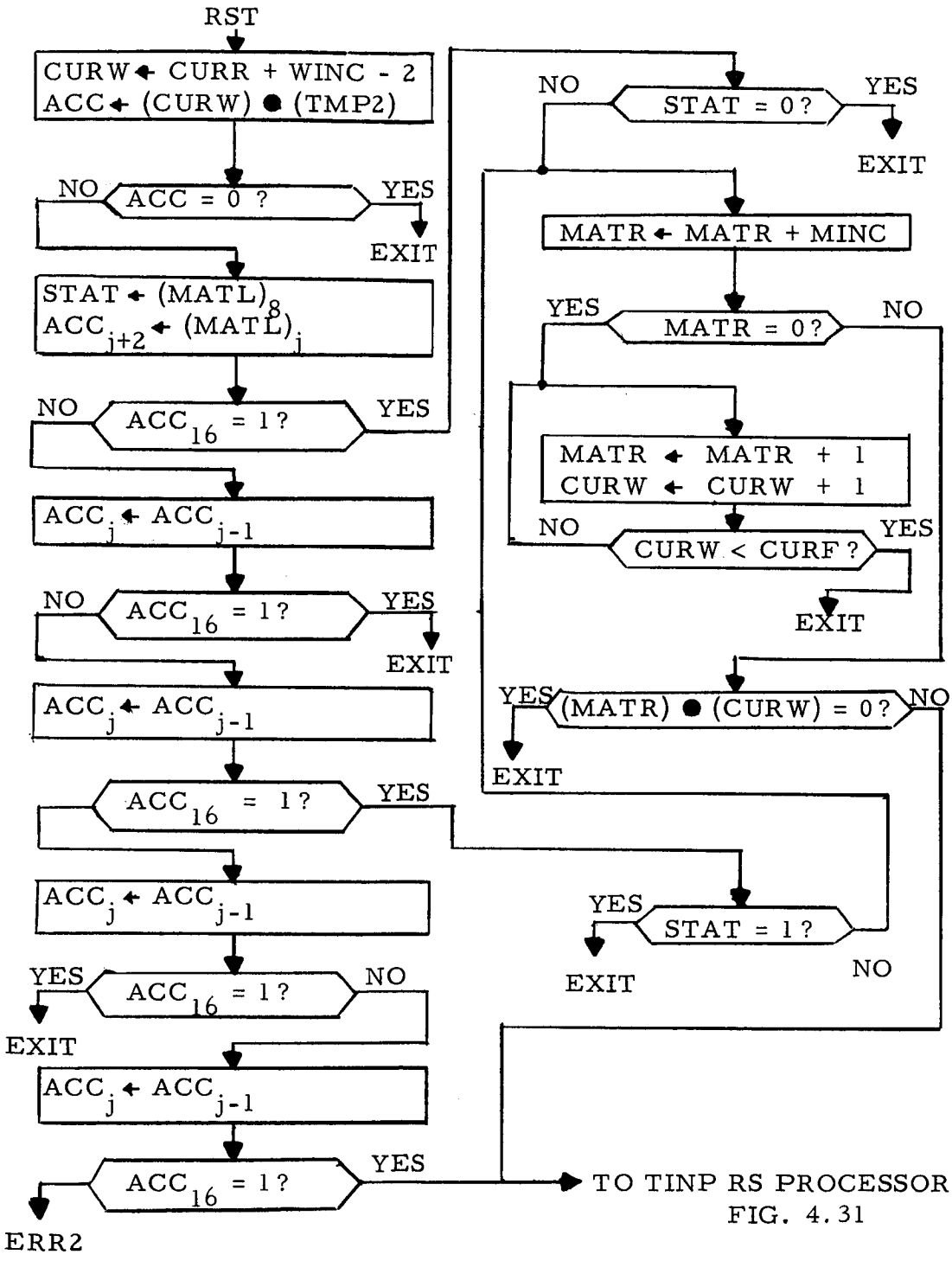


Figure D.1 RST Flip-Flop with Steering Gate
Flow Chart Modification

be the duration of the monostable pulse. Thus, the triggering of the gate part would cause an entry into CLK in the position of the corresponding P line, along with the insertion of storage word one into the clock word of the P line.

It would be desirable to have the propagation delay and the monostable pulse duration subject to statistical variations. This could be done by utilizing one of the unused bits in the P line label to indicate that the parameters in this P line are subject to statistics. This would of course require the use of a D card.

These monostable multivibrators could be designated as M elements and loaded into the computer either before or after the G cards. They would appear on the T cards and cause the number of P lines reserved in the matrix to be equal to P + M. Thus after the processing of the P cards, MATL would be augmented by M multiplied by MINC, and the G and M cards processed. As each M card was processed one of the previously skipped P lines would be used.

This would require a special label. Two possibilities come readily to mind. First the unused combination of both bits 9 and 16 of the label being 1's. A second possibility is to restrict

the number of inputs to a gate to eight or less, then bit 10 in the gate label becomes available.

It should be noted that although a single input gate was assumed in the preceding discussion, there is nothing in the processing to prevent any type of gate, AND or OR from being used. NAND and NOR are possible, but pointless since no external input is drawn off.

D.4 A GPG element

This is a rather specialized case of the DMV element discussed above. This is essentially a DMV with an input gate which contains both transient and level inputs. This could be handled by using two matrix lines for the gate portion of the multivibrator. One matrix line would be for the level input, the other for the transient input. Stimulation of the level input would produce no effect while the presence of a transient of proper polarity on the transient input would require that a 1 be present on the level input. Processing would be identical to that of the set or reset inputs in the discussion of the modified RST discussed above.

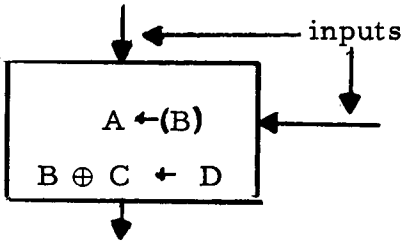
D.5 A Majority Element

This could be treated in a manner analogous to an AND GATE. A special subroutine could be created and summoned by a label code. An extra bit could be obtained by removing the redundancy among the ON, STATUS, and INVERT bits. This bit would be used in the label examination to jump to a subroutine whose function would be to scan the inputs to the element and ascertain whether a majority were 1's or not. Then, depending on the state of the element an exit could be made, a transient initiated, or a transient could be killed.

If the element is not of the simple majority type, provision would have to be made for expansion of the label to indicate somewhere the critical number of active inputs required to fire the element.

Appendix E. Flow Chart Conventions

E.1 Operation Block



(1) Regardless of where the entry point joins the block, the complete block is executed in sequence.

(2) Mnemonics enclosed by parentheses indicate indirect addressing, otherwise reference is to contents of the mnemonic.

E.2 Symbols

(1) ← replacement

(2) + addition (arithmetic)

(3) - subtraction (arithmetic)

(4) x multiplication (arithmetic)

(5) ÷ division (arithmetic)

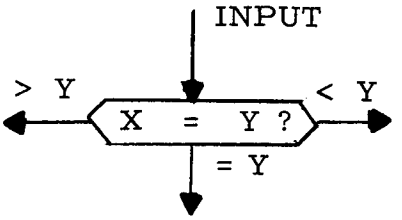
(6) ● logical and

(7) ⊕ ring sum

(8) ACC accumulator*

(9) 'XX XX are expressed in octal

E.3 Decision Block



(1) The contents are always phrased as a question.

(2) The exits are each labelled as possible answers to the question in the block.

E.4 Notes

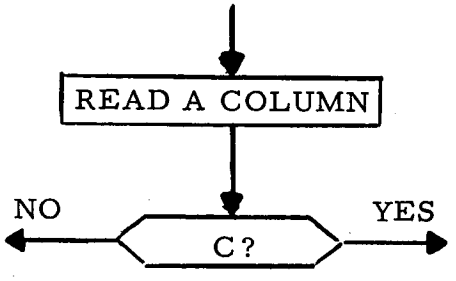
(1) Output operations are normally not detailed, merely indicated as, for example, TYPE "FLAG" which causes the ASR-33 to type out FLAG.

(2) Similarly, input operations are not normally detailed as for example READ A CARD causes the card reader to read a card.

---

*The DDP-116 contains a double accumulator for multiplication and division. Unless otherwise noted the flow charts refer to the digits of interest regardless of register location.

(3) In blocks immediately following an input instruction, the character input is referred to by its letter or numeral rather than any special input code. As an example the chart at the right asks whether the column read contains the symbol "C".



References

1. Pressman, A. I., "The Design of Transistorized Circuits for Digital Computers", John F. Rider, Inc., New York, 1959.

2. E. E. Staff of Texas Instruments, Inc., "Transistor Circuit Design", McGraw-Hill, New York, 1963.

3. Ware and Becker "Reliability vs. Component Tolerances", Proc. IRE. Vol. 51, No. 9, September, 1963.

4. H. J. Gray, "Digital Computer Engineering", Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1963.

5. Electronic Design News - June, 1965.

6. Proceedings of the IEEE, Volume 52, No. 12, December, 1964.

7. Transactions of the IRE Professional Group on Military Electronics, Volume MIL-5, No. 3, July, 1961.

8. Leonard C. Maier, "Integrated Circuits in Industrial Equipment", IEEE Spectrum, Vol. 1, No. 6, June, 1964.

9. R. M. van Slyke, "Monte Carlo Methods and the PERT Problem" ORC 63-11(RR) Operations Research Center, University of California, Berkeley, May, 1963.

10. Dr. W. Lynch - private communication.

11. Computer Control Company, Inc., "Programmers Reference Manual for the DDP-116 General Purpose Computer", Document No. 71-281. Framingham, Mass., January, 1965.

12. Computer Control Company, Inc., "DAP-116 Manual for the DDP-116 General Purpose Computer" Document No. 71-300. Framingham, Mass., January, 1965.

13. W. B. Davenport, Jr., and W. L. Root, "An Introduction to the Theory of Random Signals and Noise" McGraw-Hill New York, 1958.

14. E. C. Guilleman "Synthesis of Passive Network", J. Wiley, New York, 1957.

15. O. Tauskey and J. Todd, "Generation of Pseudo-Random Numbers" in Symposium on Monte Carlo Methods, Ed., H. A. Meyer, Wiley, New York, 1954.

16. A Ralston and H. S. Wulf "Mathematical Methods for Digital Computers" John Wiley, New York, 1960.

17. Dr. A. Lynch - Private communication.

18. R. S. Ledley, "Digital Computer and Control Engineering", McGraw-Hill, New York, 1960.