NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report 32-1542

Reliability Computation Using Fault Tree Analysis

Paul O. Chelson

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

December 1, 1971

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report 32-1542

Reliability Computation Using Fault Tree Analysis

Paul O. Chelson

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

December 1, 1971

Prepared Under Contract No. NAS 7-100
National Aeronautics and Space Administration

Preface

The work described in this report was performed by the Quality Assurance and Reliability Division of the Jet Propulsion Laboratory. The program listing for the fault tree analysis described in this report is contained in Technical Memorandum 33-512, *Program Listing for Fault Tree Analysis of JPL Technical Report 32-1542,* Jet Propulsion Laboratory, Pasadena, Calif., Dec. 1, 1971.

Contents

Contents (contd)

Figures (contd)

Abstract

A method is presented for calculating event probabilities from an arbitrary fault tree. The method includes an analytical derivation of the system equation and is not a simulation program. The method can handle systems that incorporate standby redundancy and it uses conditional probabilities for computing fault trees where the same basic failure appears in more than one fault path.

Reliability Computation Using Fault Tree Analysis

I. Introduction

This report describes a fault tree modeling technique having application to a variety of system analysis situations. The functional and mathematical methods discussed in this report are intended for use primarily in the areas of reliability and safety analysis—especially during the early design phases of a Jet Propulsion Laboratory (JPL) hardware development program. However, because of the flexibility of the fault tree technique, it can also be used on a mature design as an effective failure analysis and prevention device based on actual test and operational data inputs.

Simply defined, a "fault tree" is a graphical representation of the logic that relates certain specific events or primary failures to an ultimate undesired event.

In reliability and systems analysis, a fault tree analysis has many uses. The fault tree technique can be used to perform a reliability analysis of a particularly important undesired event—either before a failure, as a prediction method, or after a failure, as a means for finding the most likely cause of failure.[1] Fault tree techniques can also be used as a method for performing failure mode and effects analyses, system sensitivity studies, and safety analyses. (Reference 1 gives a more detailed discussion of fault tree safety analysis.)

The fault tree technique of analyzing the system, by recording combinations of events in an easy-to-read diagram, lends itself readily to the use of probability estimates for the most critical elements of the system being diagrammed. In a simple analysis this information can be discovered by inspection. In more complex systems, an analytical method is usually necessary to utilize the fault tree most effectively. One such analytical technique is to assign probabilities of failure to the different elements of the fault tree, develop an algebraic expression representing the fault tree, and then obtain a probability of undesired event occurrence for the system represented by the fault tree.

The aim of this report is twofold. The first aim is to discuss fault trees and their application and construction, including techniques for applying fault trees to systems incorporating standby redundancy. The second aim is to present a computerized method for performing the analytical reduction and computation of fault trees.

[1]This was done at JPL as part of the analysis of the Surveyor 2 failure. A fault tree was used to study the possible causes of the tumbling of the spacecraft, which was apparently caused by engine 3 of the vernier propulsion subsystem failing to ignite.

II. The Fundamentals of Fault Tree Construction

A. What Is a Fault Tree and How Is It Used?

A fault tree is a unifying analysis tool. A fault tree provides a methodology for the reliability analysis of all hardware elements of the system (mechanical and electromechanical as well as electronic), as well as all factors that degrade system reliability.

Chief among the advantages of the fault tree approach are the following

(1) The "top-down" failure mode and effects analysis lends itself to better organization and more thorough analysis than a conventional failure mode and effects analysis.

(2) The fault tree, being a basic logic diagram of system, subsystem, and component faults, is directly amenable to probabilistic computation, and, as such, serves the same purpose as a reliability model. In general, fault trees are often event-oriented, whereas reliability predictions using reliability block diagrams are usually hardware-oriented.

(3) The system faults at the top of the fault tree can be immediately related, both functionally and probabilistically, to detailed failures and problems.

(4) When used to analyze detailed assemblies, a fault tree makes obvious the interdependencies which always exist but which are frequently neglected in a reliability model.

Thus, a fault tree serves the function of a failure mode and effects analysis, while, at the same time, it provides the additional probabilistic information that a typical reliability analysis provides in the form of reliability block diagrams. Because the fault tree is a top-down approach, it can be halted at any hardware level and can be used to explore certain critical faults in greater depth than others. In summary, a fault tree offers sufficient flexibility to be practical for a number of different users—it is a "practical" failure mode analysis, as well as a single-point failure and/or reliability prediction tool.

A few negative aspects of fault trees should be mentioned. It is a "negative" or failure-oriented study of the system, unlike a reliability block diagram that considers "success" modes. Sometimes this is a disadvantage, but it is probably more frequently an advantage, since it orients the designer and the reliability engineer to thinking of eliminating "single-point failure" situations in a design. Another disadvantage is that, if the fault tree analysis is carried to the piece-part level for a large system, the detail is overwhelming. In such a situation, a compromise approach would be necessary: the use of (1) the fault tree approach for the most critical functions and (2) a chart-form failure mode and effects analysis coupled with a standard reliability analysis for the rest of the system.

B. Guidelines for Fault Tree Construction

The construction of a fault tree begins by defining the topmost undesired event one wishes to analyze. This can be almost anything: an event as gross as "astronaut's life endangered" or something more detailed, like "solar panels fail to deploy." This undesired event becomes the output event of the fault tree and is placed at the top of the tree.

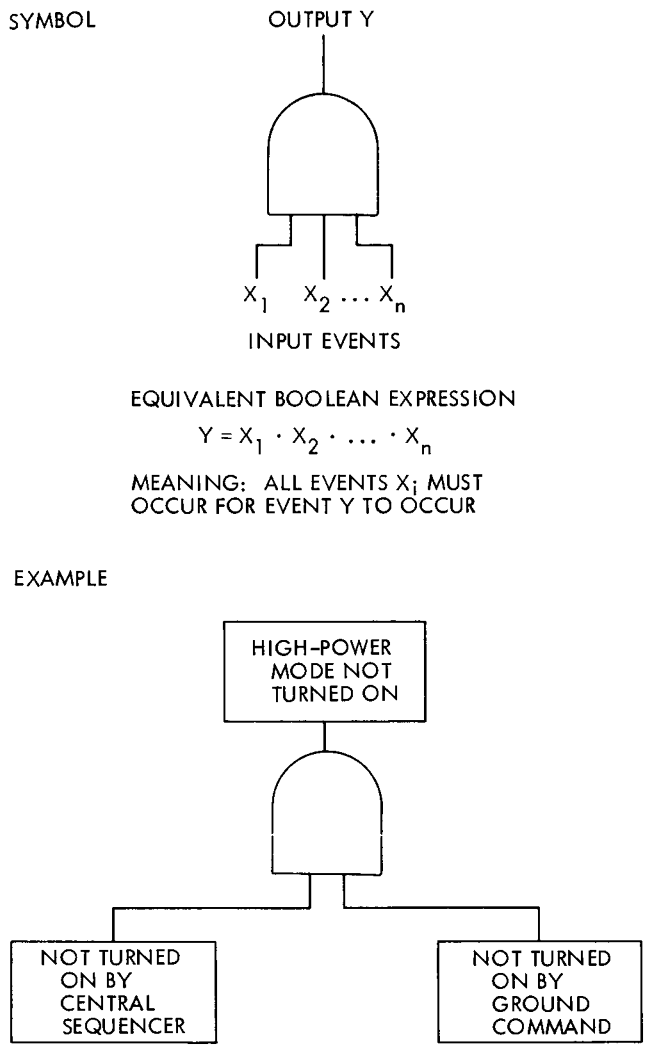
The analyst then relates events that could cause this undesired event. The events are related through the use of logic symbols. Fault trees have been in use for many years and the symbology used has become standardized, as explained below.

The two most important logic functions used in constructing fault trees are the AND gate and the OR gate. These are illustrated in Figs. 1 and 2. There are several variations of these basic functions. They are the INHIBIT gate, the PRIORITY AND gate, and the EXCLUSIVE OR gate. Because the PRIORITY AND gate and the EXCLUSIVE OR gate were found, through experience, to have little real value in fault tree construction, they were not used in the modeling technique discussed in this report.

The "INHIBIT gate," another commonly used gate, with application mainly for safety analyses, is illustrated in Fig. 3. The INHIBIT gate is generally used as an AND gate except that the input event is entered in a failure rate fashion and the side condition is described as a probability. Situations requiring the use of an INHIBIT gate do not arise very frequently when the fault tree is used as a reliability tool; thus this symbol is not used as an INHIBIT gate at JPL. Also, the program described in this report will accept either probabilities or failure rates as basic inputs, so that an INHIBIT gate can be handled as an AND gate.

With the hexagon thus freed from use as an INHIBIT gate, it can be used as a STANDBY gate in describing systems incorporating standby redundancy. The STANDBY gate is illustrated in Fig. 4.

The STANDBY gate behaves somewhat like an AND gate except that a priority is involved. The prime path must fail before the backup is considered. Thus, in the


SYMBOL
OUTPUT Y
X_1 $X_2 \cdots X_n$
INPUT EVENTS
EXAMPLE
HIGH-POWER MODE NOT TURNED ON
NOT TURNED ON BY CENTRAL SEQUENCER
NOT TURNED ON BY GROUND COMMAND

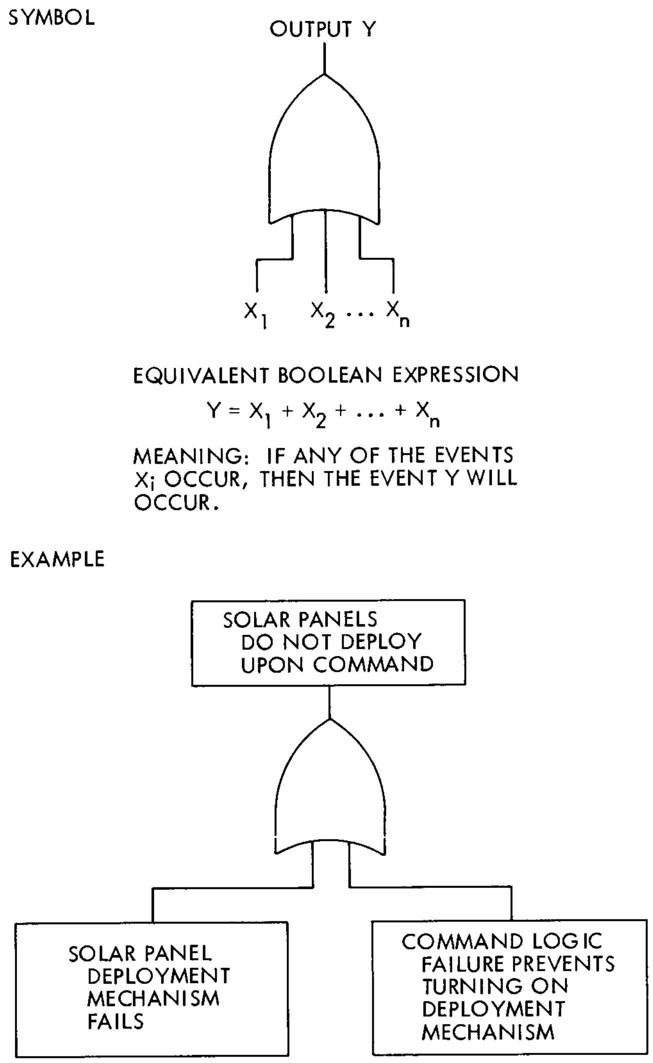
EQUIVALENT BOOLEAN EXPRESSION

$Y = X_1 \cdot X_2 \cdot \ldots \cdot X_n$

MEANING: ALL EVENTS X_i MUST OCCUR FOR EVENT Y TO OCCUR

DISCUSSION: IN THE EXAMPLE, BOTH THE CENTRAL SEQUENCER AND THE GROUND COMMAND MUST FAIL TO TURN THE HIGH-POWER MODE ON FOR THE HIGH POWER MODE NOT TO BE TURNED ON.

Fig. 1. AND gate


SYMBOL
OUTPUT Y
X_1 $X_2 \cdots X_n$
EXAMPLE
SOLAR PANELS DO NOT DEPLOY UPON COMMAND
SOLAR PANEL DEPLOYMENT MECHANISM FAILS
COMMAND LOGIC FAILURE PREVENTS TURNING ON DEPLOYMENT MECHANISM

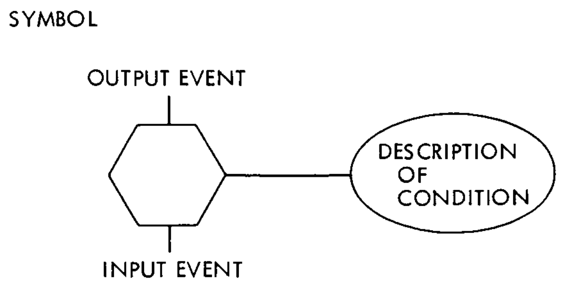
EQUIVALENT BOOLEAN EXPRESSION

$Y = X_1 + X_2 + \ldots + X_n$

MEANING: IF ANY OF THE EVENTS X_i OCCUR, THEN THE EVENT Y WILL OCCUR.

DISCUSSION: IN THE EXAMPLE, EITHER A DEPLOYMENT MECHANISM FAILURE OR A COMMAND LOGIC FAILURE WILL PREVENT THE SOLAR PANELS FROM DEPLOYING.

Fig. 2. OR gate

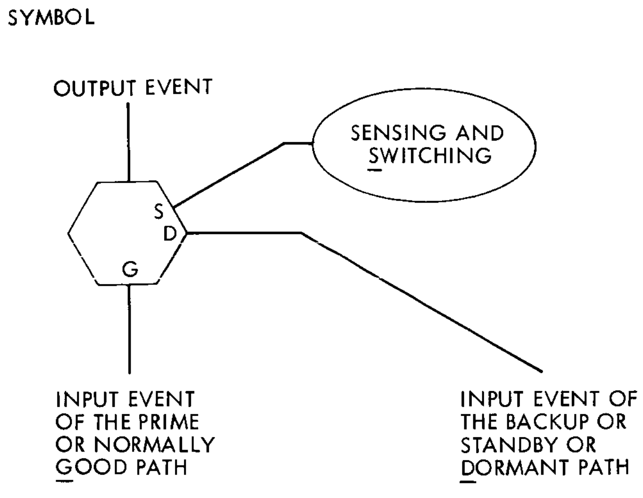


MEANING: IF THE INPUT EVENT OCCURS AND THE CONDITION IS SATISFIED, THEN THE OUTPUT EVENT WILL OCCUR.

Fig. 3. Standard INHIBIT gate—not used in this program

example involving traveling wave tube (TWT) transmitters, the output event, "transmitters fail," occurs if both the good or prime input occurs, "TWT 1 fails," *and* the backup event occurs, TWT 2 fails, or various switching events or failures will cause the output event to occur. The particulars of how the switch can be considered will be treated later—there are several options.

The above-mentioned symbols are used to represent the logic for the fault tree. There are also three standard symbols used to represent *events* in the fault tree; these are illustrated in Fig. 5. In addition to these symbols, a triangle


SYMBOL
OUTPUT EVENT
SENSING AND SWITCHING
S
D
G
INPUT EVENT OF THE PRIME OR NORMALLY GOOD PATH
INPUT EVENT OF THE BACKUP OR STANDBY OR DORMANT PATH

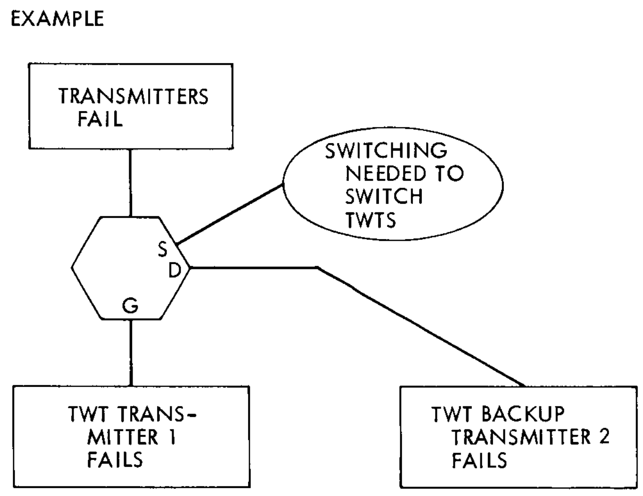

EXAMPLE
TRANSMITTERS FAIL
SWITCHING NEEDED TO SWITCH TWTS
S
D
G
TWT TRANSMITTER 1 FAILS
TWT BACKUP TRANSMITTER 2 FAILS

Fig. 4. STANDBY gate

is often used to indicate a transfer of a fault tree path to another portion of the fault tree.

The actual construction of the fault tree is accomplished by the analyst's relating the top block, through logic symbols, to specific events, then relating these events to more specific events, until the analysis is carried to the functional level desired. Each sequence of events that leads to the ultimate undesired event is called a "fault path."

Fault tree construction is probably best understood by studying examples. Figure 6 is an example of a reliability fault tree. Further examples can be found in Section IV of this report and in Ref. 1.

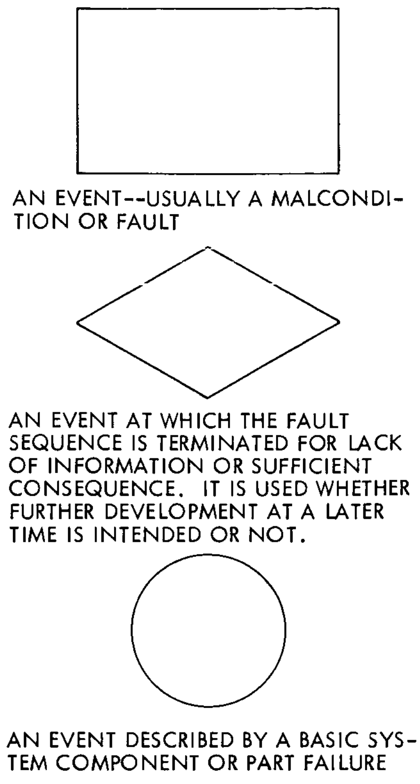

AN EVENT--USUALLY A MALCONDITION OR FAULT
AN EVENT AT WHICH THE FAULT SEQUENCE IS TERMINATED FOR LACK OF INFORMATION OR SUFFICIENT CONSEQUENCE. IT IS USED WHETHER FURTHER DEVELOPMENT AT A LATER TIME IS INTENDED OR NOT.
AN EVENT DESCRIBED BY A BASIC SYSTEM COMPONENT OR PART FAILURE

Fig. 5. Fault tree event symbols

If it is desired to perform a mathematical analysis of the fault tree, all sequences of events must be carried to basic failures, that is, events for which it is possible to obtain a numerical probability of occurrence.

This brings up a special problem. It is not always easy to get realistic and meaningful probabilities for all basic failure situations. For part and component failures, there are several publications that are useful for deriving failure rate data that are meaningful for space environments (see Refs. 2, 3, and 4). Human problems are much more difficult to predict realistically, since human errors must be considered inadvertent and, as such, are difficult to describe with numerical probabilities. Ordinarily, large-scale flight simulators are the only good means for getting such data.

As in all useful models, the problem of getting meaningful input data is of paramount importance to the fault tree validity and worth. To ensure achieving such lofty objectives, the fault tree analyst must have a practical personal knowledge of the system being analyzed and a good understanding of its relationships with interfacing subsystems or systems. This requires the analyst to have a

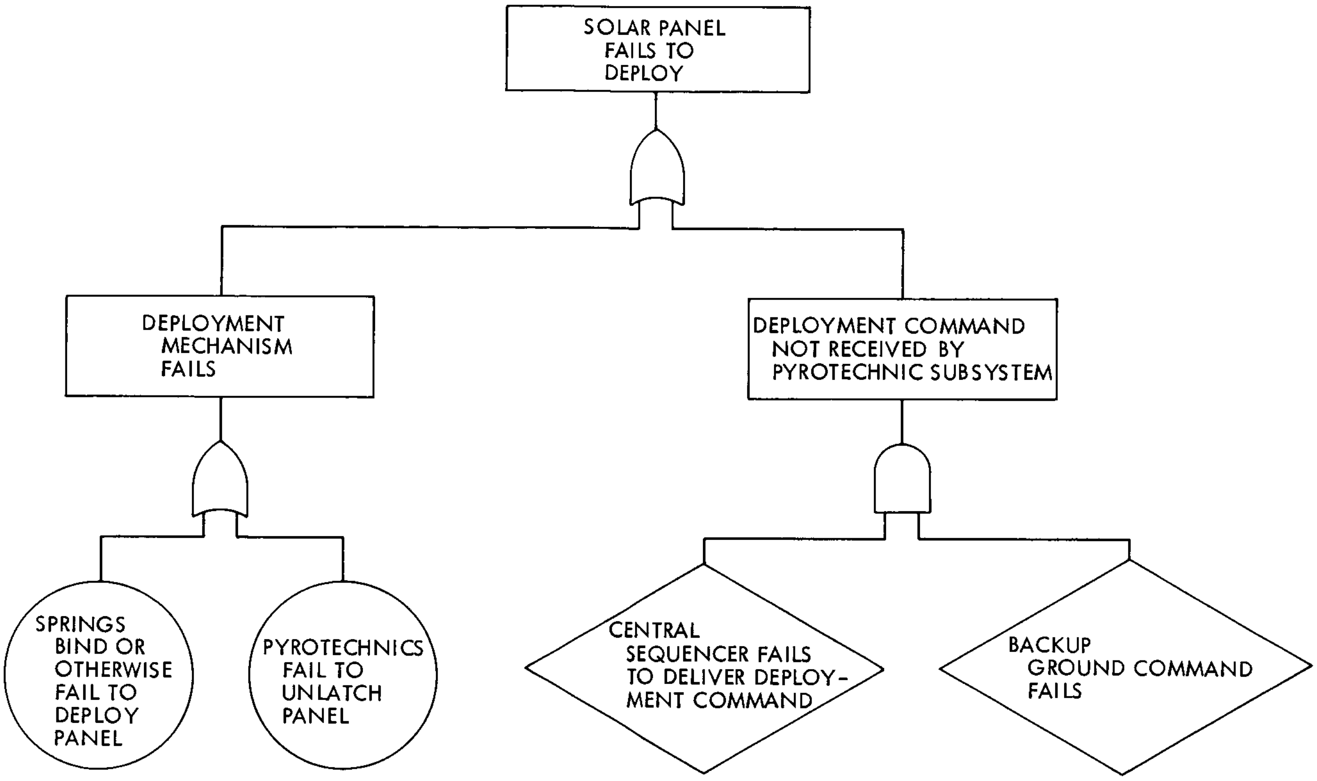

SOLAR PANEL FAILS TO DEPLOY
DEPLOYMENT MECHANISM FAILS
DEPLOYMENT COMMAND NOT RECEIVED BY PYROTECHNIC SUBSYSTEM
SPRINGS BIND OR OTHERWISE FAIL TO DEPLOY PANEL
PYROTECHNICS FAIL TO UNLATCH PANEL
CENTRAL SEQUENCER FAILS TO DELIVER DEPLOYMENT COMMAND
BACKUP GROUND COMMAND FAILS

Fig. 6. Reliability fault tree example

thorough understanding of the basic workings of the system under study, plus an ability to visualize the operation of the system in the environment it was designed and packaged for. This can be no mean feat, especially in the earlier design stages of a system's development, when many of the more important mission profile elements necessary for an accurate fault tree study turn out to be fuzzily defined. Interestingly, this type of situation exposes one of the important advantages of fault tree analysis: its flexibility in being easily modified as situations develop or goals are altered, not only with realism within the framework of the task, but also (by use of the modeling technique described in this report) within a short time.

III. Fault Tree Mathematics

A. The Monte Carlo Method vs. the Analytical Method

The mathematics of fault trees can become quite complicated in the computational sense. There are two general ways to handle fault tree mathematics: the Monte Carlo Method and the analytical method. The Monte Carlo method describes the fault tree with Boolean expressions and then simulates the fault tree on a fast, large-capacity computer using Monte Carlo techniques. One of the advantages of this method is that it allows for such things as repair time to be inserted for each input block as well as its associated failure probability. Repair time considerations are very important for modeling repairable systems, such as an aircraft system or any system that can be given periodic maintenance. An example of the use of the Monte Carlo method is described in Ref. 1.

The main disadvantage of the Monte Carlo method is the large number of simulations needed for statistical accuracy (many thousands, or, more often, millions of simulations). This, unfortunately, can be very time-consuming and expensive, even on a very fast, large-capacity computer. Also, in a normal JPL project, the opportunity to insert repair time is not a valid argument for the Monte Carlo method since, for unmanned JPL space programs, repair times are not usually relevant, because there are no man-rated maintenance situations.

Therefore, the Monte Carlo method was not chosen for the mathematical portion of the fault tree analysis. In practice, it appeared that an analytical method of modeling (by deriving the ultimate event directly as a function of the probabilities of the individual input events) would certainly be a valid technique, as well as being more practical for JPL needs. Unfortunately, a large problem with the analytical approach is that deriving the formula for the fault tree by hand is a very complicated procedure for all but the simplest of fault trees, and there were no general computer programs available for developing this formula through the use of a computer. But, since such general programs are available for reliability block diagrams (Refs. 5 and 6), it was decided to develop a companion technique for using an analytical approach to fault tree modeling.

This report will describe how, by taking advantage of the logic similarities of fault trees and reliability block diagrams, a general computer program was developed that takes the fault tree as input, gives as an output the formula describing the probability of the fault tree, and performs the calculations for the probability of system failure, using the basic known input probabilities. This program is listed in JPL Technical Memorandum 33-506 (Ref. 7) and is discussed in Section IV.

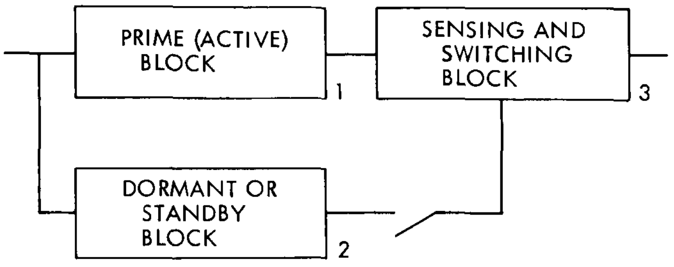
B. A Description of the Analytical Method

An important (and useful) property of fault trees is their logic similarity to reliability block diagrams. For instance, consider an AND gate (Fig. 7). Events $X_1, X_2, \ldots, X_n$ must all fail in order to cause X to fail. Therefore, letting P be probability of success and $\overline{P}$ be probability of failure, or $1 - P$, we have

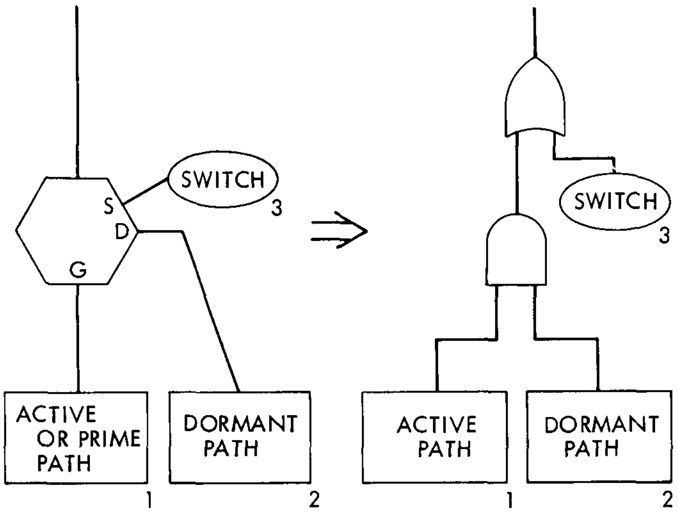
$$\overline{P}(X) = \overline{P}(X_1) \cdot \overline{P}(X_2) \cdot \ldots \cdot \overline{P}(X_n)$$

But this is also the formula for the reliability block diagram of $X_1, \ldots, X_n$ in parallel. Similarly, an OR gate can be considered as a reliability block diagram with the blocks in series. This is also illustrated in Fig. 7.

The standby gate is a more difficult gate to analyze. It is probably easiest to consider the form that we want the equivalent block diagram to be. The program (Ref. 6) that we use to compute the reliability block diagrams stores standby redundancy as follows:



The information is stored by the program by storing the fact that blocks 1 and 2 input to block 3, plus the information in another array that block 3 is a sense-switch block that controls the standby block 2. Thus, we wish to convert the information contained in a STANDBY gate to the form of the reliability block diagram above. This can be done in two simple stages. The STANDBY gate is converted to a fault tree using only AND and OR gates as follows:



The fault tree on the right is then converted to a reliability block diagram using the rules for AND and OR gates, and the resulting diagram (with the additional information that one is a sense-switch and one is a dormant block) is of the form needed. The figure above would indicate that the switch is in series with the standby system of the active and dormant path. This is not the case. It is drawn in series for convenience, but the actual behavior and effect of a failure in the sensing and switching can be varied. This will be considered later when we discuss switching options.

The rules above can be applied many times. Thus, when one has a large fault tree, a repeated application of the rules to AND, OR, and STANDBY gates will reduce the fault tree to a reliability block diagram. This is illustrated in Figs. 8 and 9.

Often in fault trees it is convenient to have the same basic fault appear in several fault paths. When this occurs, it can not be handled directly as it was above—an additional point must be considered. When a basic failure occurs more than once in a fault tree, it will also appear more than once in the equivalent reliability block diagram. The physical meaning of the block's appearing for multiple

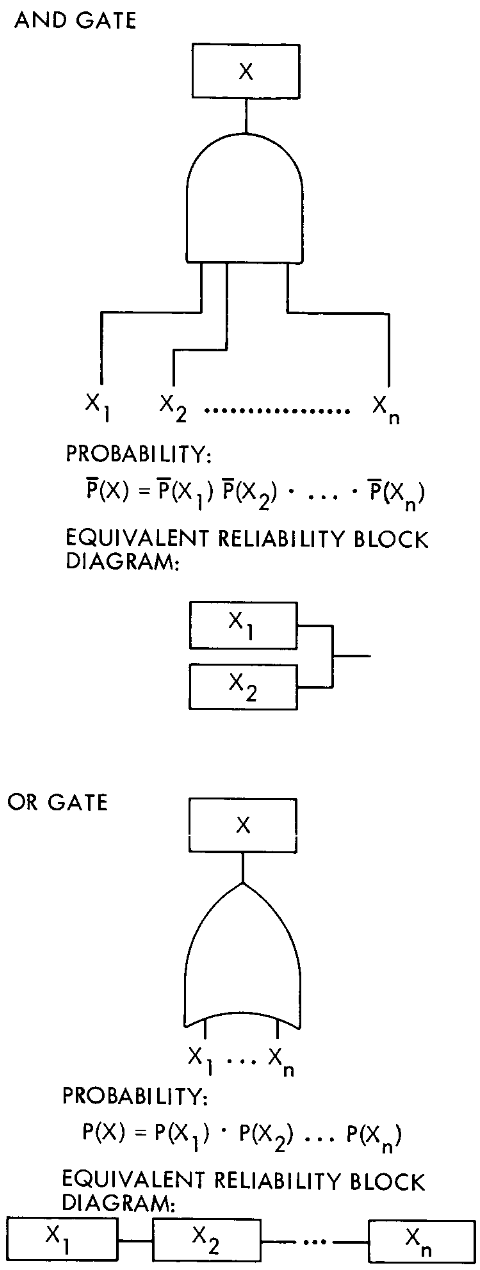

AND GATE
X
X_1 X_2 .... X_n
PROBABILITY:
$\bar{P}(X) = \bar{P}(X_1)\,\bar{P}(X_2) \cdot \ldots \cdot \bar{P}(X_n)$
EQUIVALENT RELIABILITY BLOCK DIAGRAM:
X_1
X_2
OR GATE
X
$X_1 \ldots X_n$
PROBABILITY:
$P(X) = P(X_1) \cdot P(X_2) \ldots P(X_n)$
EQUIVALENT RELIABILITY BLOCK DIAGRAM:
X_1 X_2 ... X_n

Fig. 7. Similarities of fault trees and reliability block diagrams

times is that the same piece of hardware appears in several success paths for the convenience of describing the success relationships. When such blocks appear several times, they are called "equivalent" blocks, and must not be treated as independent blocks. Since the program used for computing the system reliability can handle most arrangements of "equivalent" blocks, this does not present a problem.

Thus, "equivalent" blocks occur in the reliability block diagram when a basic failure appears more than once in the fault tree. There are a few restrictions in this program as to

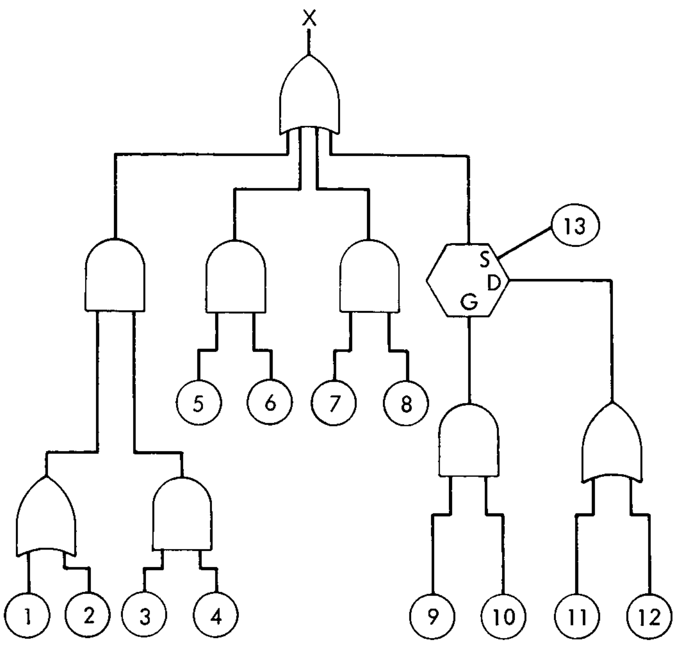


Fig. 8. An example of a fault tree. Low-level faults 1–12 are related to the system-level fault X

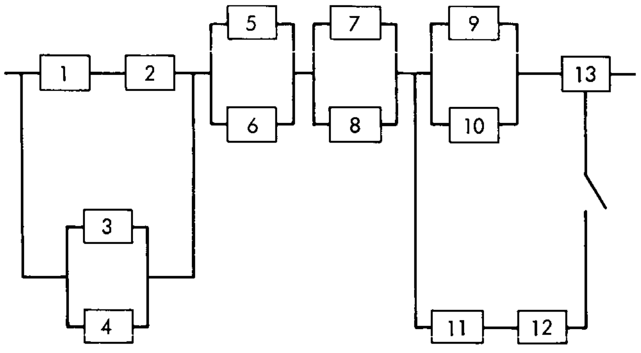


Fig. 9. The equivalent reliability block diagram for the fault tree of Fig. 8

when the same basic failure can, or cannot, start more than one fault path. These restrictions are as follows:

(1) If the same basic failure appears multiply, but only in the active portion of the fault tree (i.e., not under a STANDBY gate), there are no restrictions.

(2) If a basic failure appears under a STANDBY gate, then if it appears multiply, all other occurrences must be in the same branch of the STANDBY gate. Thus, one can have the same failure appearing several times in the active (prime) or standby (dormant) portion of the tree, but not in both. This second restriction is undesirable. However, if a situation like this arises, it is often possible to avoid it by modifying the fault tree slightly.[2]

Thus, a fault tree that uses the logic symbols AND, OR, and STANDBY can be converted to a reliability block diagram as described above. Computer programs able to handle reliability block diagrams have already been developed, so if we can develop a program to convert fault trees to block diagrams, then we can completely handle the mathematics of fault trees by the use of a computer.

IV. The Fault Tree Computer Program

A. Objectives

The objective of this effort was to produce a computer program to:

(1) Analyze fault trees which use AND, OR, or STANDBY gates to describe their logic.

(2) Allow the fault tree to be entered into the computer in as simple a format as possible.

(3) Include in the printout enough information so that it is possible to write out the formula that is used to calculate the probability for the fault tree.

(4) Be able to show the sensitivity of system failure to individual block unreliability by varying the probabilities for the individual blocks during execution of the program.

B. Methods

The computer program written to satisfy the objectives above is in three parts. The first part converts the fault tree to a reliability block diagram, the second part analyzes the reliability block diagram, and the third part computes the probability. The second part, which uses the probability tree method, is an adaptation of the program discussed in Ref. 6. The details of the second and third parts will not be discussed here since they are discussed in detail in Ref. 6. The program was written in FORTRAN V to run on a Univac 1108, Exec 8 system.

[2]It should be noted that the Monte Carlo method does not have this restriction. However, this seems a minor inconvenience when one compares the cost and accuracy of this method to those of the Monte Carlo technique.

C. Program Limitations

To accommodate the storage capacity of the Univac 1108, the following limitations were assumed:

(1) The fault tree can have at most 49 input blocks (basic failures). This is not a serious restriction. If the fault tree is larger, it is only necessary to break it into portions, compute the portions, and then enter each portion as one input block on another run. It is important then that each portion is independent of each other.

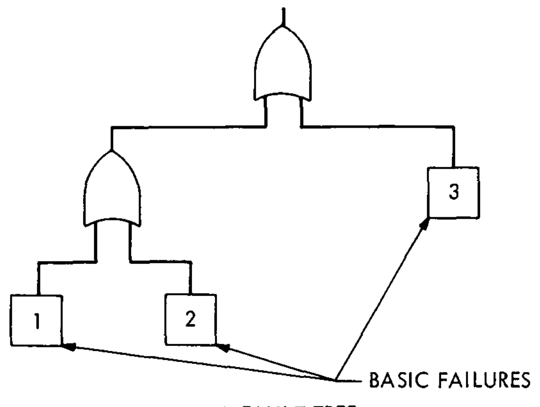
(2) There can be at most 14 inputs to any AND or OR gate. This means 14 real inputs, which means 14 inputs after the fault tree has been reduced to its simplest terms. This is illustrated in Fig. 10. This problem might not be obvious when the fault tree is entered, since the tree need not be entered in its reduced form. However, the program reduces the fault tree to its simplest terms. Thus, if a gate has more than 14 inputs, an error message is printed. If this should happen, it is simple to break the diagram into smaller portions and proceed as in (1).

(3) There can be at most 20 logic gates in any fault path. If more than this should occur, reduce the fault tree by combining consecutive AND or OR gates, or if this cannot be done, break the fault tree into portions and proceed as in (1). If dormancy is used in the tree, the maximum number of logic gates in the fault path for the dormant elements is 19.

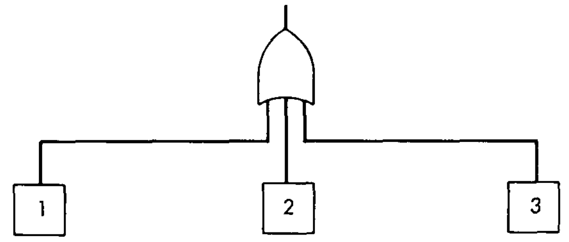
(4) There can be at most 200 success paths when the equivalent reliability block diagram is analyzed. The number of success paths depends on the complexity of the fault tree. Since it is hard to know this ahead of time, an error message is printed if more than 200 paths exist. If this should happen, it is simple to break the fault tree into smaller portions and proceed as in (1).

D. Input

The fault tree input is made as simple as possible. To enter a tree, arbitrarily number each input block (basic faults) with numbers between 1 and 49. Each block must have a different number. Number the AND gates with A01, A02, etc., the OR gates with O01, O02, etc. For STANDBY gates describing standby redundancy, it is important to know which input is which, and so the active, or normally good, input to the STANDBY gate is numbered with G01, G02, etc.; the standby, or dormant, input with D01, D02, etc. The input describing the switch used to switch in the



A FAULT TREE



THE SAME FAULT TREE REDUCED TO SIMPLEST TERMS

Fig. 10. Reduction of a fault tree to its simplest form

backup path when the path entering the G input of a STANDBY gate fails is entered as a switch with S01, S02, etc.

Assume there are n input blocks in the fault tree. The input cards will now be described.

1. Fault paths. The *fault paths* (i.e., the description of the fault tree) are entered. the first n cards list the n fault paths in the following form (with one card per fault path):

(1) Columns [1–2],[3] the block number. Format (I2).

(2) Columns [3–5], [6–8], . . . [60–62], identify all the gates that make up the fault path which starts with the block number punched in columns [1–2], listed from the bottom to the top. It is necessary to use one letter and two digits to identify each gate. (The FORTRAN format for entering the gate information is 20A3, the first alphanumeric character is the letter describing the gate (A for AND, O for OR, G, D, S for the normally good, dormant, and switch inputs to STANDBY). The next two characters identify the number of the gate.)

(3) The nth card must have a 1 in column [80] to indicate that this is the end of the description of the fault paths.

(4) If there are equivalent blocks or the same basic failure beginning more than one fault path, use the same number to begin each fault path. Thus, if "13" describes a failure block that begins three separate fault paths, there will be three "13" cards, one for each of the fault paths beginning at "13." (In this case there will then be $n + 2$ fault path cards.)

2. Mission time. This card contains the failure distribution information (1 = exponential, which is the only option currently available) and mission time (total number of hours). The card should be as follows: Column [1] blank, a 1 in [2], mission time [3–14] (E12.7 format). For example, exponential distribution with mission time = 100,000 h would have the following in columns [1–14]:

```
bl+.1000000+06          b = Blank in column one.
```

3. Active parameters (failure rates and probabilities). This section contains one data card for each block number for all failures except failures in switch branches of STANDBY gates, with the block's initial reliability R_0 (which is 1–Q_0, where Q_0 is the initial failure probability for the block) and failure rate (lambda). The format for these cards is block number [1–2], lambda [3–14] (E12.7 format), R_0 [15–24] (F10.7 format). If [15–24] are left blank, R_0 is set equal to 1.0 by the program. The data card for the last block number in this series has a 5 in [80].

4. Dormant parameters. If there are no STANDBY gates, Subsections 4 and 5 are omitted. This section contains provisions for assigning a dormant failure rate (lambda dormant) to each failure block in a dormant path of a STANDBY gate. There are three options:

(1) No failures occurring in dormancy—insert a blank card.

(2) Read in a dormancy factor on one card (blank [1–2], dormancy factor [3–14] (E12.7 format)), which the program will multiply by the active lambda of each standby block to yield the dormant lambdas for each standby block.

(3) Read the lambda dormant for each standby block individually. To do this, the first card will have a 99 in [1–2], blank [3–80]. Then read one data card for each standby block, with standby block [1–2] and lambda dormant [3–14] (E12.7 format). The data card for the last standby block in this series has a 6 in [80].

[3]Numbers in brackets indicate column numbers.

5. Switching options. This section describes the data input for each switch entering a STANDBY gate. Designate one of the four switching options (0, 1, 2, 3) with input cards as follows:

0 = Perfect switching (probability of switch working = 1.0). Sense switch number [1–2], blank [3–79], 0 in [80]. One input card per switch.

1 = Constant probability that switch works. Sense switch number [1–2], blank [3–14], (probability [15–24] (F10.7 format), 1 in [80]. One input card per switch.

2 = Dormant failure rate only for switch. Sense switch number [1–2], lambda dormant [3–14] (E12.7 format), 2 in [80]. One input card.

3 = Dormant and active failure rate for switch (2 cards per switch). Sense switch number [1–2], lambda dormant [3–14] (E12.7 format), 3 in [80]. Sense switch number [1–2], lambda active [3–14] (E12.7 format), 3 in [80].

If there is no standby redundancy, no switch cards are needed.

6. Recalculation option card. This card is blank [1–79] and has the last column [80] set equal to 7, 8, or 9, as follows:

7 = Recalculate the diagram with the new parameters. The program will loop back and start reading from section 2 (mission time) through this section. This permits varying mission time, R_0, active/dormant lambdas, and switch options.

8 = Read in a new fault tree and new parameters. The program will loop back and start reading from section 1 (fault path inputs). This permits varying fault tree configurations.

9 = End of computer run.

For a better understanding of the steps necessary for entering the fault tree into the computer, the example of Fig. 6 is redrawn labeling all gates and input events that are necessary for the computer input (Fig. 11). Figure 12 then shows the format that would be used to enter this fault tree.

Another example, this one more complicated, is illustrated in Figs. 13 and 14. This example is the fault tree of Fig. 8.

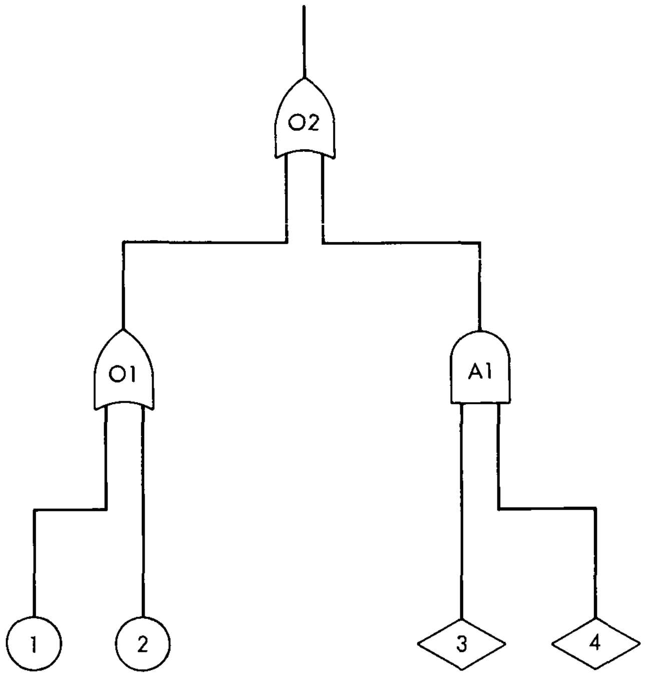


Fig. 11. Example of Fig. 6, redrawn ready for input to the computer program

E. Output Controls

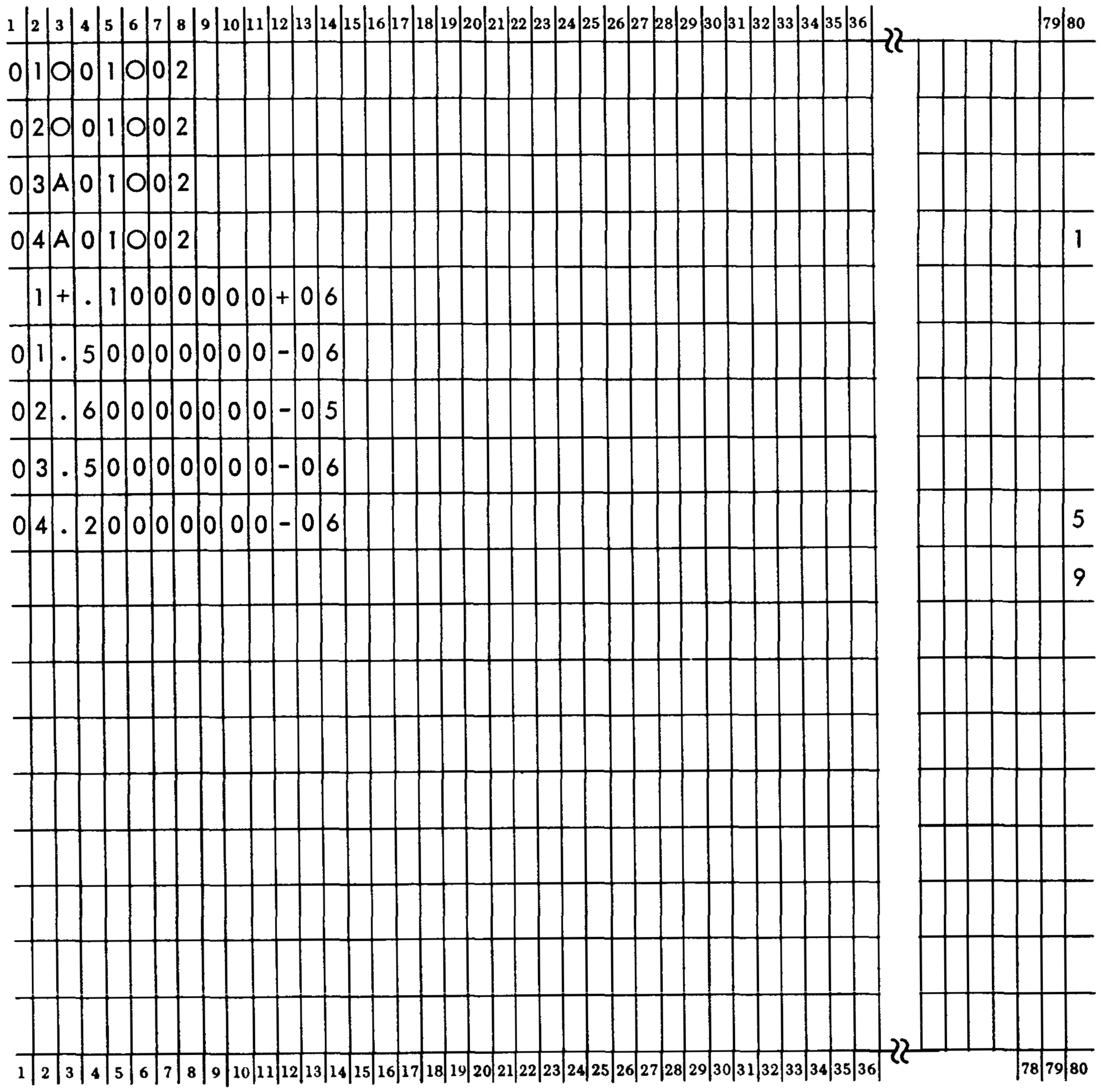
The computer output can be controlled by certain variables. Some of the output is dependent on a program variable called IPRINT.

The IPRINT variable is normally set by the program to be 0. When it is set to 0, only the fault tree (Pages 1 and 2 in Section F), the parameters, and the result (Page 5 in Section F) are printed. This is the output usually desired by the user and nothing need be done by the user for this option. It is, however, possible to override this by making the very first card of the data deck (the card that follows the @ XQT card) the following: IPRINT = n in the first 8 columns, where n = 1, 2, 3, or 4 for the following print options:

Option 1. In addition to the printout from option 0, the equivalent reliability block diagram as well as any modifications to the fault tree by the program are printed. This is described as Pages 3 and 4 in Section F.

Option 2. Option 1 plus the overall probability trees developed to analyze the reliability block diagram.

Option 3. Option 2 plus all probability trees used by the program.



```
01O01O02
02O01O02
03A01O02
04A01O02                                                                       1
 1+.1000000+06
01.50000000-06
02.60000000-05
03.50000000-06
04.20000000-06                                                                 5
                                                                               9
```

Fig. 12. Input cards for fault tree of Fig. 11

Option 4. Option 3 plus diagnostic information.

If one wishes to use options 2 or 3, the printout obtained is described in Ref. 6 as well as very briefly below as Pages 4a, 4b, and 4c. The output from option 4 is of use only to someone following through the program listing in great detail. The pages of printout are described in Subsection F.

In addition to IPRINT, there is another variable that is normally set by the program but can be overridden by the user if desired. This option is controlled by the program variable NSIG and refers to the number of "significant" digits that will be printed for the computed system reliabilities. The term "significant" digits is defined in a very special way. The "significant" digits are the non-nine digits in the reliability number. Thus 0.99986, 0.975, and 0.52 all have two "significant" digits. The value of NSIG is set equal to 3 by the program, and so three "significant" digits will be printed for the system reliability unless NSIG is specified as something else by the user. To do this, columns 9–15 of the first card of the data deck (the same card that contains the IPRINT specification if IPRINT is also being specified) contains

,NSIG = n where $n = 1, 2, \ldots,$ or 8.

Thus if both IPRINT and NSIG are being specified by the user, the first card of the data deck should look like the following in columns 1–15:

IPRINT = 2, NSIG = 4

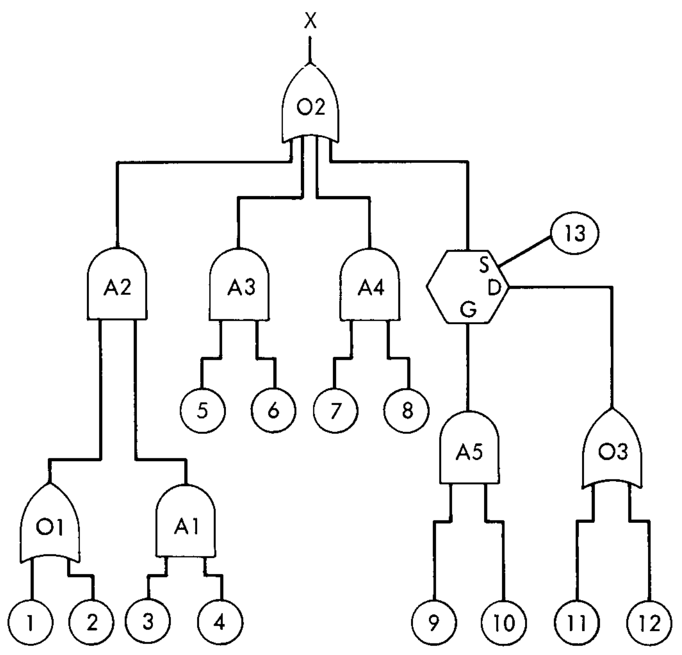


Fig. 13. Figure 8 redrawn, numbering the gates for input to the computer program

F. Detailed Program Output

1. **Page 1.** The first page lists any sets of equivalent blocks that have been assigned by the program. This occurs when the same basic failure block (which was input by using the same number to start the fault path) occurs in several paths. An example might look like the following:

```
EQUIVALENT BLOCKS
      1=  7
```

The numbers 1 and 7 are the same fault, but are now listed as separate faults, i.e. as different numbers, in the fault tree and in the block diagram. However, when one evaluates the system equation for the probability for the system, the fact that 1 and 7 are the same fault is taken into account by use of conditional probabilities.

2. **Page 2.** The fault tree is printed as a check for the user to be sure that the fault tree that was input was the one desired. If there are equivalent blocks, they are listed with separate numbers. Also, any switch blocks (i.e., switch inputs to STANDBY gates) will be listed with the dormant blocks that they control. Typically, Page 2 looks like the following, which is the output for the fault tree described in Figs. 6, 11, and 12:

```
 FAULT TREE PATHS
 BLOCK  1   001 002
 BLOCK  2   001 002
 BLOCK  3   A01 002
 BLOCK  4   A01 002
```

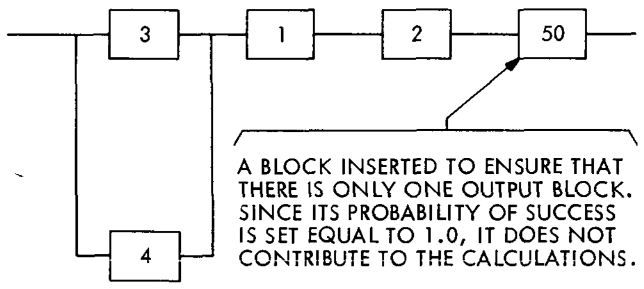
3. **Page 3.** This page is printed only if IPRINT ≠ 0. It contains the modified fault tree and is usually needed only for diagnostic purposes. This page contains a listing of the fault paths with successive AND and OR gates removed, as well as with STANDBY gates converted to the equivalent AND, OR relationship described in Subsection III-B. Note that these new AND and OR gates are denoted by D and S respectively. Typical modified fault tree paths are shown below:

```
          MODIFIED FAULT TREE PATHS
 BLOCK  1   002
 BLOCK  2   002
 BLOCK  3   A01 002
 BLOCK  4   A01 002
```

4. **Page 4.** Page 4 contains the equivalent reliability block diagram for the fault tree being analyzed. Any equivalent blocks or sense switches are listed. The reliability block diagram is presented by listing the inputs and outputs of all blocks. It is easy to draw the block diagram from this listing. This page, which is printed only if IPRINT ≠ 0, typically looks like the following:

```
RELIABILITY BLOCK DIAGRAM

BLOCK  1    INPUT   3    4
            OUTPUT  2

BLOCK  2    INPUT   1
            OUTPUT  50

BLOCK  3    INPUT
            OUTPUT  1

BLOCK  4    INPUT
            OUTPUT  1

BLOCK 50    INPUT   2
            OUTPUT
```

Since this listing identifies each block with its inputs and outputs, a unique reliability block diagram is described. This block diagram, when drawn out, looks like the following:


3
1
2
50
4
A BLOCK INSERTED TO ENSURE THAT THERE IS ONLY ONE OUTPUT BLOCK. SINCE ITS PROBABILITY OF SUCCESS IS SET EQUAL TO 1.0, IT DOES NOT CONTRIBUTE TO THE CALCULATIONS.

5. **Page 5.** Page 5 lists for each original block its active and dormant failure rate R_0 unless $R_0 = 1.0$, and its reliability unless it is a standby block replaced by a pseudoblock (which are numbered between 51–65). In the latter case, the reliability of the pseudoblock 51-65 will be printed. Each sense switch will have its active and dormant failure rate and probability printed. Then, the mission time and the reliability of the system are printed. This printed page typically looks like the example in Fig. 15.

6. **Other pages.** In addition, several other pages are printed after Page 4 and before Page 5 if IPRINT = 2 is set. These contain information about the probability trees used to analyze the block diagram. A more detailed discussion of this information is contained in Ref. 6, where reliability block diagrams are discussed. These pages are as follows:

(1) Page 4a. This lists the success paths of the probability tree. Plus numbers indicate a success, while minus numbers indicate a failure. Each path of the probability tree is indicated by up to two lines of print, consisting of up to 50 numbers. Note that this information about the probability tree is not needed by the casual user. It is from this tree that one can derive the system reliability equation that is used in the computation phase of the program. This is also true for Pages 5 and 6 listed below. The output page containing the probability tree would typically look like the following:

```
PROBABILITY TREE   50

 50   2   1   3

 50   2   1  -3   4
```

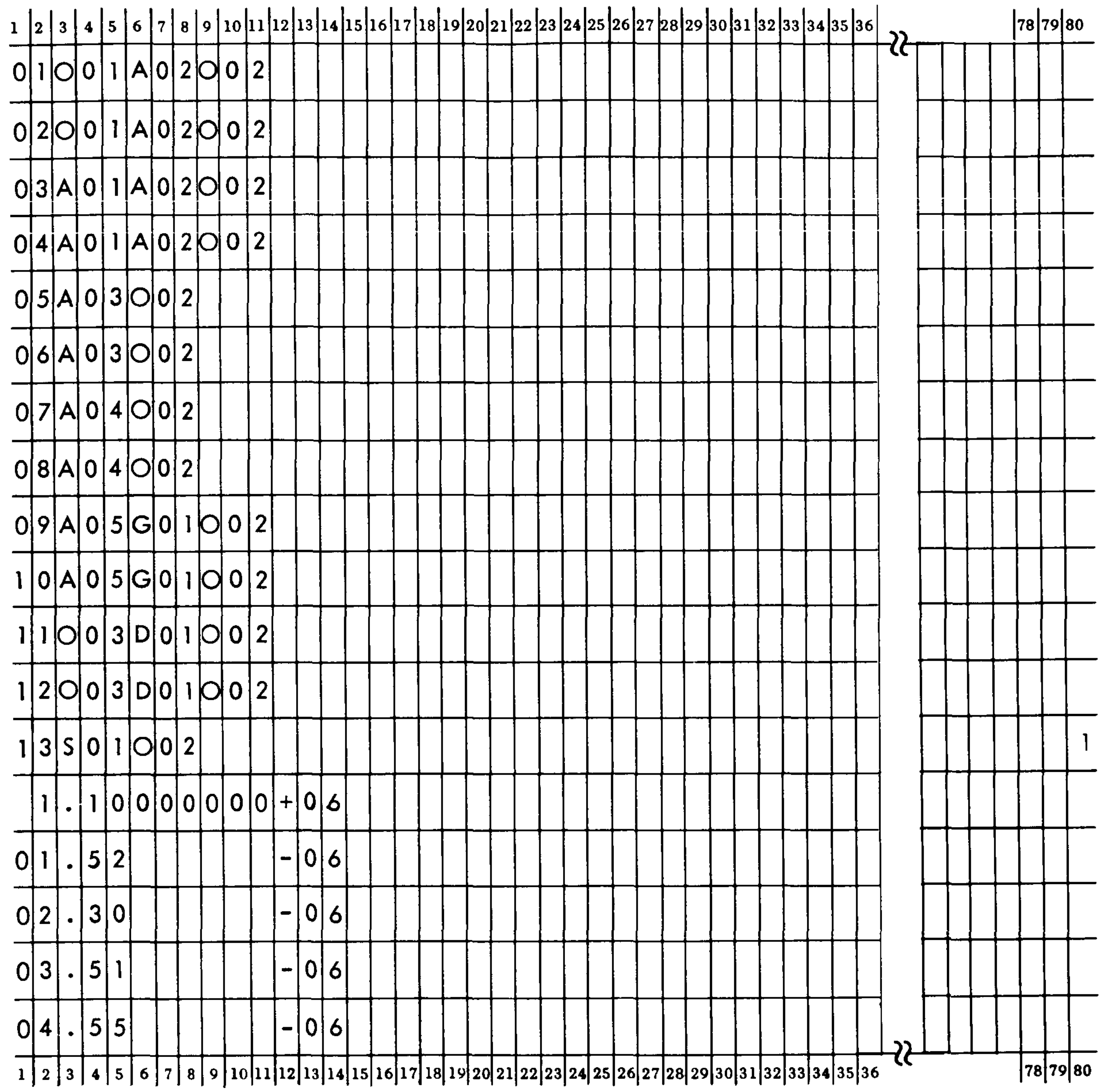
This is the probability tree for the original block diagram. If there is no standby (no sense switches) then the system reliability equation can be derived from this tree. This is done by taking the product of the probabilities in each success path and summing this over all the success paths. The equation for the example shown above would be

$$\left(p_{sys} = p_{50} \cdot p_2 \cdot p_1 \cdot p_3 + p_{50} \cdot p_2 \cdot p_1 \cdot q_3 \cdot p_4\right)$$

(2) Page 4b. When standby redundancy is present, pseudoblocks are often needed when there is more than one block in the dormant path. These pseudoblocks are designated with numbers 51–65. All places where pseudoblocks are used are listed along with the standby probability trees that they replace (the success routes of the standby system).

(3) Page 4c. If Page 4b is printed, Page 4c is printed also. It lists the original probability tree (Page 2) with standby blocks (51-65) replacing their respective standby trees. This replacement usually causes a reduction of the number of original success paths.

Pages 6 and 7 are printed only if IPRINT >2. They are generally not of interest to the user. They are needed, however, to actually derive the system reliability equation when standby redundancy is



```
01O01A02O02
02O01A02O02
03A01A02O02
04A01A02O02
05A03O02
06A03O02
07A04O02
08A04O02
09A05G01O02
10A05G01O02
11O03D01O02
12O03D01O02
13S01O02                                                                       1
 1.10000000+06
01.52      -06
02.30      -06
03.51      -06
04.55      -06
```

Fig. 14. The input for the fault tree of Fig. 13

present. The terms P_i and $\overline{R}$ can be derived from this information. These terms are described in Ref. 6, and this would be needed to fully understand the information on these pages.

7. Some examples. The outputs are given (for the normal IPRINT = 0) option for the two examples whose inputs were illustrated. The output for the example of Figs. 6, 11, and 12 was shown in Subsection E-5. The output for the example of Figs. 13 and 14 is shown in Fig. 16.

V. Summary

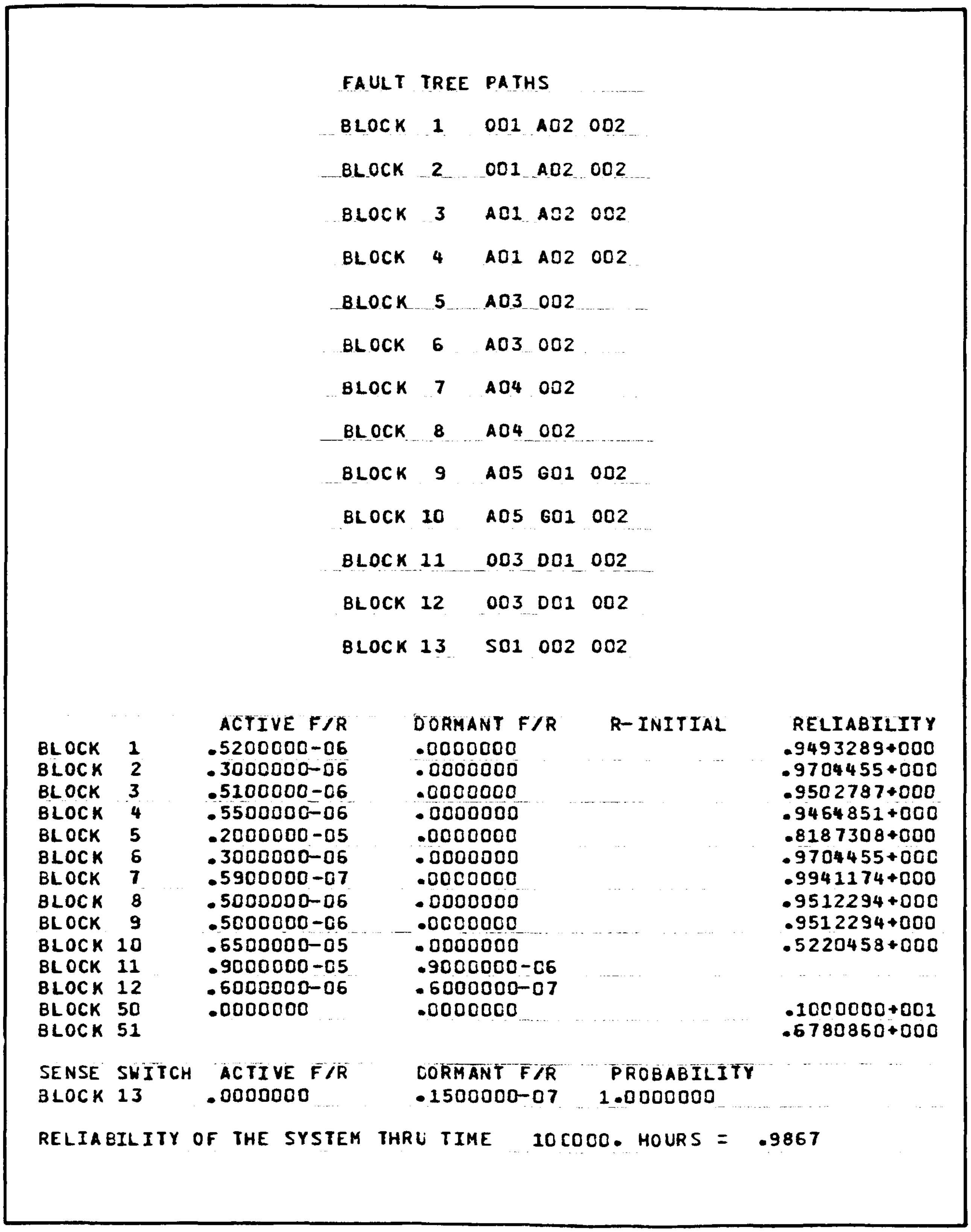
Fault tree analysis is a powerful reliability analysis tool as well as a safety analysis tool. This report has described a method (and a computer program) that allows one to obtain a prediction from the fault tree. This method includes analytical derivation of the probability equation for the undesired event being analyzed; it is not a simulation. Conditional probabilities make it possible to compute fault trees in which the same basic failure ends more than one fault path.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	78	79	80
0	5	.	2	0							-	0	5																									
0	6	.	3								-	0	6																									
0	7	.	5	9							-	0	7																									
0	8	.	5								-	0	6																									
0	9	.	5								-	0	6																									
1	0	.	6	5							-	0	5																									
1	1	.	9	0							-	0	5																									
1	2	.	6								-	0	6																									5
		.	1																																			
1	3	.	1	5							-	0	7																									2
																																						9

Fig. 14 (contd)

```
                    ACTIVE F/R        DORMANT F/R      R-INITIAL          RELIABILITY
BLOCK   1         .5000000-06        .0000000                             .9512294+000
BLOCK   2         .6000000-05        .0000000                             .5488116+000
BLOCK   3         .5000000-06        .0000000                             .9512294+000
BLOCK   4         .2000000-06        .0000000                             .9801987+000
BLOCK  50         .0000000           .0000000                             .1000000+001

RELIABILITY OF THE SYSTEM THRU TIME    100000. HOURS =   .522
```

Fig. 15. Typical example, Page 5 output



```
                        FAULT TREE PATHS

                        BLOCK  1    001 A02 002

                        BLOCK  2    001 A02 002

                        BLOCK  3    A01 A02 002

                        BLOCK  4    A01 A02 002

                        BLOCK  5    A03 002

                        BLOCK  6    A03 002

                        BLOCK  7    A04 002

                        BLOCK  8    A04 002

                        BLOCK  9    A05 G01 002

                        BLOCK 10    A05 G01 002

                        BLOCK 11    003 D01 002

                        BLOCK 12    003 D01 002

                        BLOCK 13    S01 002 002


                 ACTIVE F/R       DORMANT F/R      R-INITIAL       RELIABILITY
BLOCK   1      .5200000-06       .0000000                         .9493289+000
BLOCK   2      .3000000-06       .0000000                         .9704455+000
BLOCK   3      .5100000-06       .0000000                         .9502787+000
BLOCK   4      .5500000-06       .0000000                         .9464851+000
BLOCK   5      .2000000-05       .0000000                         .8187308+000
BLOCK   6      .3000000-06       .0000000                         .9704455+000
BLOCK   7      .5900000-07       .0000000                         .9941174+000
BLOCK   8      .5000000-06       .0000000                         .9512294+000
BLOCK   9      .5000000-06       .0000000                         .9512294+000
BLOCK  10      .6500000-05       .0000000                         .5220458+000
BLOCK  11      .9000000-05       .9000000-06
BLOCK  12      .6000000-06       .6000000-07
BLOCK  50      .0000000          .0000000                         .1000000+001
BLOCK  51                                                         .6780860+000

SENSE SWITCH   ACTIVE F/R        DORMANT F/R     PROBABILITY
BLOCK 13       .0000000          .1500000-07    1.0000000

RELIABILITY OF THE SYSTEM THRU TIME   100000. HOURS =  .9867
```

Fig. 16. The output for the example of Figs. 13 and 14

References

1. Feutz, R.J., and Waldeck, T.A., "The Application of Fault Tree Analysis to Dynamic Systems," *System Safety Symposium*, sponsored by the University of Washington and The Boeing Company, Seattle, Wash., July 8-9, 1965.

2. Wright, F.H., *Failure Rate Computations Based on Mariner Mars 1964 Spacecraft Data*, Technical Report 32-1036. Jet Propulsion Laboratory, Pasadena, Calif., Jan. 15, 1967.

3. Wright, F.H., *Failure Rate Analysis of Mariner Venus 67 Spacecraft Data*, Technical Report 32-1266. Jet Propulsion Laboratory, Pasadena, Calif., June 1, 1969.

4. Chelson, P.O., *Failure Rate Computations Based on Mariner Mars 1969 Spacecraft Data*, Technical Report 32-1544. Jet Propulsion Laboratory, Pasadena, Calif., Dec. 1, 1971.

5. Chelson, P.O., *Reliability Math Modeling Using the Digital Computer*, Technical Report 32-1089. Jet Propulsion Laboratory, Pasadena, Calif., Apr. 15, 1967.

6. Chelson, P.O., and Eckstein, R.E., *Reliability Computation From Reliability Block Diagrams*, Technical Report 32-1543. Jet Propulsion Laboratory, Pasadena, Calif., Dec. 1, 1971.

7. Chelson, P.O., *Program Listing for Fault Tree Analysis of JPL Technical Report 32-1542*, Technical Memorandum 33-512. Jet Propulsion Laboratory, Pasadena, Calif., Dec. 1, 1971.

NASA — JPL — Coml., L.A., Calif.