THE UNIVERSITY OF MICHIGAN

INDUSTRY PROGRAM OF THE COLLEGE OF ENGINEERING

THE RELIABILITY PROBLEM IN DIGITAL COMPUTER NETS

William L. Kilmer

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in the
University of Michigan
1958

December 1958

IP-344

PREFACE

To date, most of the work in digital computer theory has been directed towards one of two ends. One of these ends has been the specification of procedures for arriving at irredundant logical designs of various types of automata. The other has been a definition of the logical capacity of automata structures in general. This has left the study of the reliability of computing machines in a badly underdeveloped state.

Until just recently, the study of methods for improving the reliability of computers was largely confined to the development of techniques for obtaining greater individual component reliabilities. The type of results that have been obtained in this area are ideal as far as the production of more reliable computers are concerned. Nevertheless, a theory of reliability based on logical design criteria has a complementary value which it alone can provide. The reason for this is as follows: It frequently happens for certain applications that more reliable computers are needed then existing components can provide. In such cases if satisfactory logical methods are not available to sufficiently improve the computer's reliability, the designer must wait for further results from physical research. And even after this, he might be told that practical reliability ultimates have already been reached for usable component types, which might be few in number to begin with, due to stringent speed, power, and loading requirements. Hence, it is easy to conceive of cases where logical design methods of improving computer reliabilities provide unique solutions to practical problems.

The study which follows pertains to one aspect of the practical type of theory which is needed. It concerns methods of specifying the design of idealized, logical nets such that the outputs of these nets are correct provided no more than any set of k or fewer component failures occur within them while the output of the net is being determined. The failures can be either transient or permanent, and the upper bounds on the associated probabilities are based on information which has appeared in the literature on the reliabilities of typical present-day computer components. The theory is nearly complete with respect to the definition of k-correcting redundant net forms which are most economical in the sense defined. The components which have been allowed are the familiar "and," "or," and "not" elements. They have been idealized to operate in discrete time with either 1 or 0 units time delay.

Bibliographic references are given throughout the text by the expression "(n)." "(n)" means consult the nth entry in the Bibliography. All abbreviations and symbols found in the text, with the pages they first appear on, are listed just before the Introduction.

The author wishes to express his gratitude to his committee members, and Professor Scott, his chairman, in particular, for their helpful remarks during his discussions of this work with them. As the recipient of a Bell Telephone Laboratories Communications Fellowship from June, 1957, to February, 1958, he is indebted to that organization for financial support over the period of the grant. Some of the work was also carried on at IBM Research Center in the summers of 1957 and 1958.

W. L. K.
Dept. of Elect. Eng.
Univ. of Michigan
September, 1958

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Number and Title con't   page

LIST OF TABLES

PAGE OF FIRST APPEARANCE OF IMPORTANT DEFINITIONS, SYMBOLS AND ABBREVIATIONS

Definition, Symbol, or Abbreviation	page
" v "	6
" · "	6
cost	8
irredundant net	8
cyclic net	8
noncyclic net	8
gnc-net	9
nc-net	9
c-net	9
k-degree failure	11
kf	11
kc net	12
kd net	12
P measure	15
binary output channel	23
individual output	23,35
realize (a logical function)	23
complete input set	23
quality (of a signal)	25

Definition, Symbol, or Abbreviation con't — page

INTRODUCTION

Thus far it seems that only few really notable works have appeared in the literature on the subject of logical design methods for improving reliability in digital computer nets, and only a small proportion of these works have dealt with the problem at the component level. Two outstanding ones that have are, "Probabilistic Logic," (5) by von Neumann, and "The Synthesis of Reliable Relay Circuits Using Less Reliable Relays," (14) by Moore and Shannon. Von Neumann's principal result is that by carrying single bits of information over large enough bundles of wires, one can synthesize arbitrarily reliable stroke organs from large numbers of individual stroke components. His technique also applies to several other kinds of unilateral components. However, its greatest defect, as far as design engineers are concerned, is the tremendous multiplication in equipment that accompanies its use.

Moore and Shannon show that by using special nets of relays in place of individual components, nearly ideal relay characteristics can be obtained. They use the bilateral property of their components to excellent advantage in order to obtain results which, economically speaking, are incomparably better than von Neumann's. Unfortunately, however, all the relay-type components which are now in existence are too slow to be of much use in modern computer construction.

A few special logical tricks have been worked out for improving computer reliabilities at the component level in certain situations. But the theory which supports them is so intuitive that is is impossible to appraise them out of their original contexts. Consequently, they are not composable into a unified procedure for obtaining maximum reliability as a function of computer cost and operating speed. Hence, the art of computer design remains without a really applicable theory for inserting component redundancy in digital computer nets to make them more reliable.

The present study is a first attempt to remedy the situation. It is based on a general set of idealized components which operate in discrete time with zero or one unit time delays. The study concerns methods of expanding the logical design of several types of irredundant computer nets so that sets of k or fewer component failures within them do not cause their outputs to be incorrect. Nets having this correcting property are said to be k-degree failure correcting, or kc. It is shown that in almost all cases of current interest, the k-degree failure correcting improvement criterion is synonymous with that of probability of net failure.

The second chapter defines a form which is probably most economical for kc nets of the type that do not contain any feedback loops within them. The third chapter derives a corresponding definition for memory nets. The fourth chapter defines 2 alternate forms of over-all kc computer nets, one of which, in each particular case, is probably most economical in the sense indicated. The second form of Chapter 4 makes extensive use of the feedback principle. Various logical nets in this form are forced to keep repeating their output determinations until no failures occur to corrupt them. The fifth chapter gives some results pertaining to most economical k-degree failure detecting nets of the type that do not contain any feedback loops in them. The last chapter contains a few comments on the nature of the results obtained in the study.

CHAPTER 1

DEFINITION OF THE PROBLEM AND A CLARIFICATION OF OBJECTIVES

1.1 DEFINITION OF THE NET COMPONENTS

All of the logical operations that present-day digital computer packages are capable of performing have the following characteristic in common: They are easily expressible in terms of the three propositional calculus functionals "and," "inclusive or," and "not." This set of functionals seems to be the most convenient one to use in expressing all the basic logical operations that have been implemented in digital computers so far. Thus, a corresponding set of idealized components has been chosen as part of the basis of the reliability study of this text.

The graphic representation of the three component types is given in Fig. 1. These components are binary in the sense that the

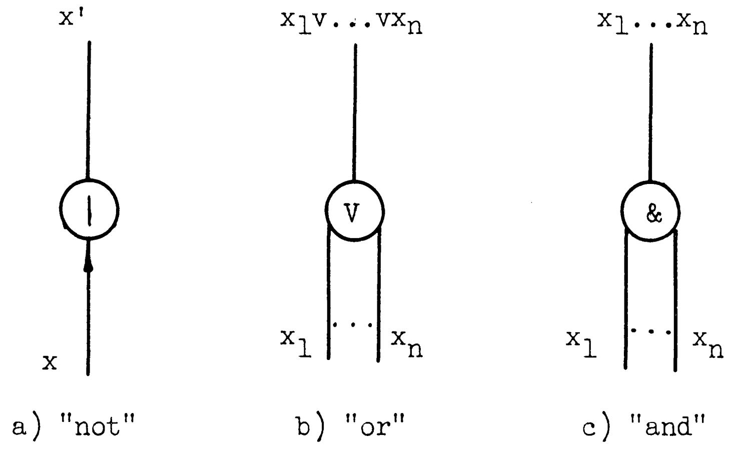

Figure 1. Representation of the Three Component Types.

state of each of their input and output wires is always either 0 or 1. The "or" and "and" components in Fig. 1b) and c) respectively, can have any finite number of inputs. The output of the "or" component is 1 if and only if one of the corresponding inputs is 1. The "or" function is written in Fig. 1b) as $x_1 \vee x_2 \vee \ldots \vee x_n$. The output of the "and" component is 1 if and only if all of the corresponding inputs are 1. The "and" function is written in Fig. 1c) as $x_1 . x_2 \ldots x_n$. The output corresponding to a 1 or 0 input to the "not" component is 0 or 1 respectively. This relation is represented in Fig. 1a) by writing x' as the negation of x.

In the study which follows, time is always discrete. That is, the complete history of a net, which is composed of interconnected sets of components of the type in Fig. 1, can be given for the interval from time 0 to some time T time units later by listing the input and output states of all the components in the net at the times 0,1, 2, . . . , T. The outputs of all the "and" and "or" components of the net are produced one time unit after the corresponding inputs arrive. But the outputs of all the "not" components are produced at the same instant that the corresponding inputs are presented. That is, there is no delay associated with the "not" component. The reason for this is that with conventional computer packages, the negation can usually be made available as quickly as the assertion. The extra cost which is usually involved is accounted for in the present theory by indicating negations with component operators.

Associated with each component is a probability of failure, denoted by e. e is the probability that a component fails transiently or permanently at time t, or has failed permanently before t. A component is said to fail transiently at time t if its output at that time is incorrect with respect to the corresponding input. A component is said to fail permanently at time t if its output at that time is incorrect with respect to the corresponding input, and if this output remains at the same state throughout all later time regardless of the component's inputs. It is assumed that the probability of failure of each component in a net of interconnected components is independent of the probabilities of failure of every other component in the net.

Abstract permanent failures may correspond to physical failures such as broken resistors and shorted tubes, since such failures cause the circuit packages in which they are located to completely ignore their inputs after they occur. Abstract transient failures may correspond to transient malfunctions in transistors, tubes, etc., which occur after these elements have deteriorated enough to begin operating marginally. Abstract transient and permanent failures are the only types that are considered in this paper.

1.2. INTERCONNECTABILITY CONVENTIONS

Any finite number of the three types of components in Fig. 1 can be interconnected to form nets according to the following rules:

No two "not" components can be connected in series. The output wire from any component can be split into a finite number of branch wires. Each of these branch wires can be used as either an input to another component or an output of the over-all net in which it is located. Output wires of different components cannot normally be merged. However, in Chapter 5 merging is permitted under certain special restrictions. These restrictions are defined there.

All nets are assumed to be synchronous. This convention brings the discrete time domains for all components common to the same net into coincidence.

The cost of a net is defined as the number of component inputs contained in the net. This cost criterion is justified to the extent that in many cases it quite closely represents the cost of the physical equipment which would be needed to physically realize the abstract net.

An irredundant net is one which realizes a given logical function, or performs a given storage function, or both, with minimum cost. It will be assumed hereafter that all nets which are ever presented for reliability improvement are irredundant.

Every net must be either cyclic or noncyclic. A noncyclic net is roughly one which does not contain any feedback paths in it. More precisely, it is defined as one which can be completely partitioned into n disjoint sets of components, called levels, as follows: The 1st level consists of all those components which receive all their inputs from the inputs to the over-all net. The jth level, $j > 1$, consists of all those components which receive at least one input from the outputs of the j-1st level, and which receive all their remaining inputs from the outputs of levels i, $i = j-2, \ldots, 1$, and the inputs to the over-all net. This completes the definition by induction.

Next, a graded net is defined. Suppose there is given a noncyclic net. Replace each of its "not" components by a wire which connects the component's previous input and output wires. Then if the resulting net can be completely partitioned into disjoint sets of components called stages as follows, it is a graded net. The 1st stage consists of all those components which receive all their inputs from the inputs to the over-all net. The jth stage, $j > 1$, consists of all those components which receive all their inputs from outputs of components of the j-1st stage. This completes the definition.

Obviously, graded nets are special types of noncyclic nets. Hereafter for notational convenience, "graded noncyclic nets" will be denoted "gnc-nets"; "noncyclic nets", "nc-nets"; and "cyclic nets", "c-nets". Figs. 2, 3, and 4 show examples of gnc-, nc-, and c-nets respectively.

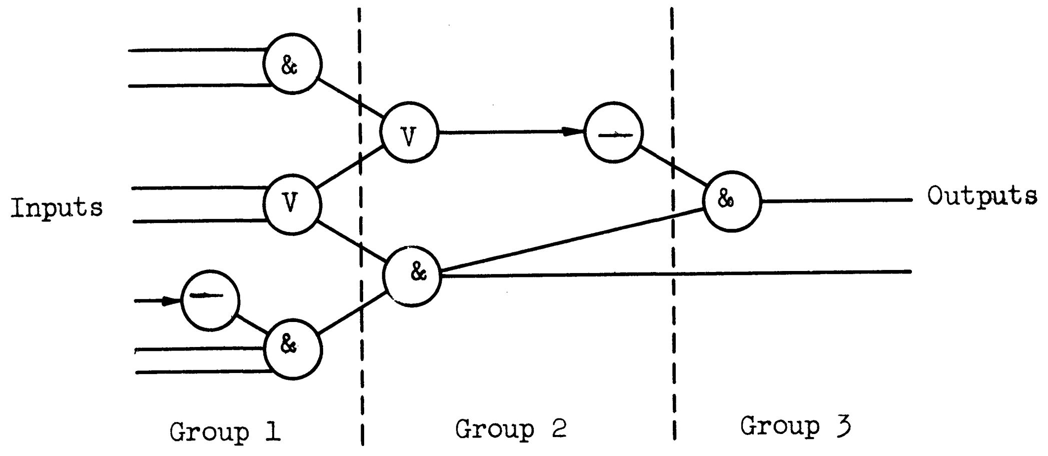

Figure 2. A gnc-net.

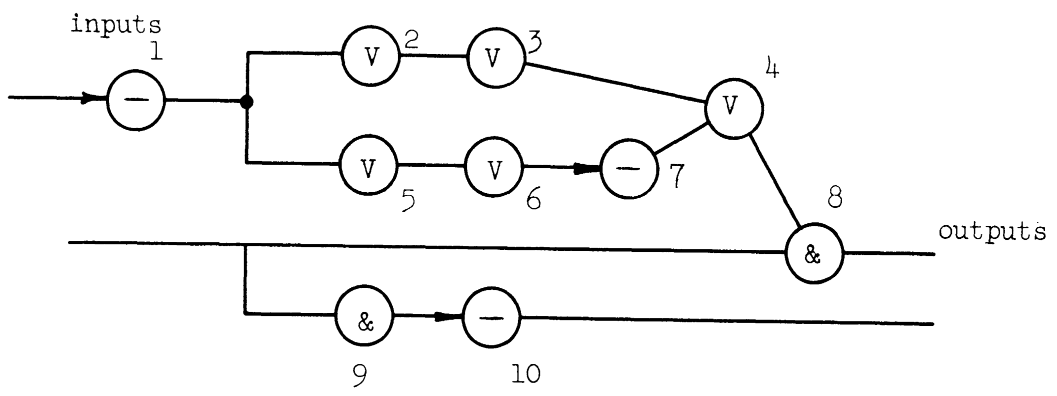

Figure 3. A nc-net.

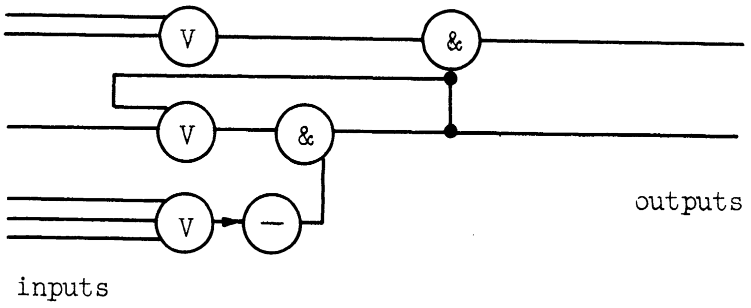

Figure 4. A c-net.

Note that if an input is presented to a gnc-net at time t, its corresponding effect in the net can be traced by observing the outputs of the first stage and associated "not" components at t + 1, the 2nd stage and associated "not" components at t + 2, etc. Also the effects of one input will occur on each output wire at only one time instant. In general there are no corresponding statements that can be made about nc-and c-nets.

1.3 SOME FAILURE DEFINITIONS

Now define groups (of components) for gnc-and nc-nets as follows: In a gnc-net, let the 1st stage and all the "not" components which have their inputs or outputs connected to this stage comprise the 1st group. Let the jth group, $j > 1$, consist of the jth stage and all the "not" components which have their inputs coming from outputs of the jth stage. In gnc-nets, the groups are disjoint. Fig. 2 shows groups for an exemplary gnc-net. In a nc-net, first replace each "not" component by a wire connecting its previous input and output wires. Then partition the resulting net into sets (of components) as follows: The 1st set consists of all components which have at least one input from the inputs to the over-all net. The jth set, $j > 1$, consists of all components which have at least one input from the outputs of the j-1st set. Hence, in general, these sets are not disjoint. Now, place the "not" components back into the net as they were originally. Then the 1st group consists of the 1st set and all the "not" components which have all their inputs or outputs connected to this set. The jth group, $j > 1$, consists of the jth set and all the "not" components which have their inputs coming from outputs of the jth set. Thus, in Fig. 3, components 1, 2, 5, 8, 9, and 10 make up group 1; components 3, 6, and 7 group 2; component 4, group 3; and component 8, group 4.

Now a <u>k-degree failure,</u> hereafter denoted <u>kf,</u> will be defined for nc-nets. Suppose an input is presented to a nc-net at t. Then if the sum of all the failures in group 1 at $t+1$, in group 2 at $t+2, \ldots,$ in the last group, (say the nth group) at $t+n$, is equal to k, a kf is said to have occurred with respect to the input time, t.

A <u>net is said to fail</u> if it produces an incorrect output. An <u>incorrect output</u> is one which is neither correct nor indicates that any failures occurred during its determination.

A nc-net is said to be a k-degree failure correcting noncyclic net, hereafter abbreviated a kc-net, if its output corresponding to an input at t is correct according to a fixed criterion, provided no more than a kf (i.e., a 1f, 2f, ..., or kf) occurs with respect to its input time, t. A nc-net is said to be a k-degree failure detecting noncyclic net, hereafter abbreviated a kd net, if its output corresponding to an input at t is either correct or positively indicates the occurrence of a jf, $0 < j \leq k$, according to a fixed criterion provided no more than a kf occurs with respect to its input time t.

If a gnc-net is kc, it must be able to produce correct outputs even if k of its components have permanently failed. On the other hand, there exist nc-, non gnc-nets which have certain single components that can in themselves cause a kf for any finite k. For example, in Fig. 3, component 8 can cause a 2f, since it belongs to both groups 1 and 3.

1.4 OBJECT OF THE STUDY

Roughly speaking, the object of the following study is to define some logical design procedures for improving the reliability of computer nets. More precisely, the object is to: 1) define economical kc and kd nets from their irredundant counterparts; 2) define economical memory nets that are failure correcting in some compatible sense; and 3) use the net forms defined in 1) and 2) to define two over-all designs of computer nets. Each of these designs is to be defined so as to be decomposable into a memory subnet and a nc-subnet such that both subnets are "k correcting" in some satisfactory sense. One of the designs is to have maximum equipment-domain redundancy and the other maximum time-domain redundancy.

1.5 JUSTIFICATION OF THE OBJECT OF THE STUDY

The expressed object is partially justifiable from the point of view that there are times when one would like a particular design which could be manufactured with up to k bad components and still operate correctly. For example: as components tend towards infinitesimal size, it might be more feasible to assemble very large numbers of fairly unreliable components to realize a logical function than to assemble a minimal number of extremely reliable ones for the same purpose.

The object is also justifiable in that it defines a conceptually simple approach to the reliability problem for the type of computer nets under consideration. That is, as the object has been stated, it gives the broad problem some character. Of course before the present approach can be accepted, it must be determined under what conditions the corresponding objective is the same as the ultimate objective of the study: namely, that of most economically decreasing the probabilities of failure of given nets.

As a first step in matching the two objectives, let us consider an example. This example has been included to illustrate some aspects and difficulties that are associated with a computation of the probability that a net will fail. Fig. 5 shows a nc-net which will correctly give an output of 1 at time t if either 1) x is 0 and y is 1 at t-2, or 2) x is 1 and y is 0 at t-2. The probability that this net will give the correct output if 1) is true is

$$(1.5.1)\qquad (1-e)^3\left[(1-e)^2 + (1-e)(2e) + e^2\right] + 4e(1-e)^2\left[(1-e)e\right] + e^2(1-e)\left[(1-e)^2 + e^2\right].$$

The first term in (1.5.1) is the probability that the components marked 3, 4, and 5 in Fig. 5 will all operate correctly to produce the correct output; the second term is the probability that one failure will occur between 3 and 4, a single compensating failure will occur between 1 and 2, and 5 will operate correctly; and the third term is the probability that two failures will occur among 3, 4, and 5 causing their incorrect affects to cancel.

The probability that the net in Fig. 5 will give the correct output if the 2) input is present is also given by (1.5.1). However with other input conditions, different expressions are needed to define the probability that the correct outputs will correspond. Hence, in order to calculate the over-all probability that the net will fail at a particular time t, the following must be done: The probability density function which describes the relative frequency of occurrence of each possible input to the net must be made available. Then the probability expression corresponding to (1.5.1) for each of the inputs must be multiplied into the probability density function, and the result integrated over the range of inputs.

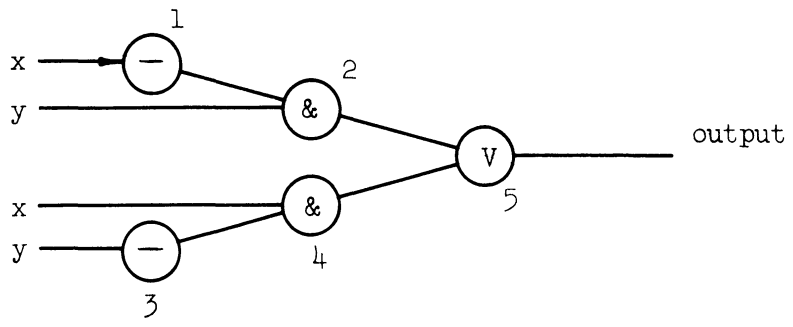

Figure 5. A nc-net.

From this it is apparent that to try to calculate the probability "that a general nc-net will fail" as a function of the net's unknown input density function is almost hopeless.

Therefore it seems imperative that the input density function of a nc-net be assumed flat in calculating the net's probability of failure. Consequently, from now on the true reliability measure of a nc-net will be considered to be the following P measure: the P for a nc-net, N, is the average probability that if an input, I_j, is presented to N at t, a kf, for some k, will occur such that N's corresponding output is incorrect according to a fixed criterion. Here, the average is taken over all I_j, where each I_j is assumed equally likely. If N has a P of 1, it is perfectly unreliable, and if its P is 0, it is perfectly reliable. Note with respect to this definition that even though the assumption of a flat input density function is not a very good one for some cases, P does afford a fairly decent measure of the confidence that one should have that the output of a net will be correct.

Now let us return to the problem of determining when the ideal and stated objectives of the study are the same. It was stated that the latter objective is useful in that it defines a conceptually simple approach to the reliability problem. The ideal objective has been approximated by defining a P measure to supplant the true reliability measure of a net. The purpose of the approximation was to obtain a calculable measure of reliability. Hence, at this point there are two precise questions which must be answered: 1) What are the conditions which must hold in order that adding components to an irredundant nc-net in order to make it kc or kd, $k > 0$, does not actually increase the P of the net? and 2) what are the conditions which must hold in order that continuously increasing k in most economically deriving kc and kd nets is the same as continuously decreasing P in most economically deriving nets with lower P values?

To answer the first question, suppose a nc-net, N, with n inputs, is given for reliability improvement. Denote by m the maximum degree of failure which can occur in N. Then suppose that redundant components are added to N to make it kc. Let this new nc-net be denoted by N'. Also denote by m' the maximum degree of failure which can occur in N'.

Now the P of N is greater than $e(1-e)^{m-1}$. Denote this quantity P_N. The P of N' is less than or equal to $(C^{m'}_{k+1}\, e^{k+1}(1-e)^{m'-k-1} + C^{m'}_{k+2}\, e^{k+2}(1-e)^{m'-k-2} + \ldots + e^{m'})$. Denote the quantity on the right side of the inequality by $P_{N'}$. Then for N' to have a smaller P than N, $P_{N'} < P_N$ must hold. If $m' << e^{-1}$, to a very good approximation this condition becomes

(1.5.2) $\quad C^{m'}_{k+1} e^{k+1} (1-e)^{m'-k-1} < e(1-e)^{m-1}$.

This in turn is closely approximated by

(1.5.3) $\quad C^{m'}_{k+1} e^{k} < 1$.

Now since $m' << e^{-1}$, Then $C^{m'}_{k+1} e^{k} < (m'-1)m'e/2 < 1$. Thus (1.5.3) can be replaced by

(1.5.4) $\quad (m'-1)m'e/2 < 1$.

This is approximately

(1.5.5) $\quad m' < (2e^{-1})^{1/2}$

Now if (1.5.5) holds and $m' << e^{-1}$, the P of N is greater than the P of N'.

Thus, a very loose set of bounds are given in partial answer to question 1) above by

Result I: If redundant components are added to an irredundant nc-net to make it kc or kd, $k > 0$, the P of the redundant net is less than the P of the irredundant net to within a very good approximation at least if $m' << e^{-1}$ and $m' < (2e^{-1})^{1/2}$ are true, where m' is the maximum degree of failure that can occur in the redundant net.

In order to answer the second question above, suppose a nc-net, N, is presented for reliability improvement. As before, denote by m the maximum degree of failure which can occur in N. Then suppose that redundant components are added to N to make it kc, (or kd) for any $k > 0$. Further, suppose that this addition is made with least possible cost, c(k).

Denote the number of different such additions that might have been made to form the new nc-net, N', as a(k). Also let m' represent the maximum degree of failure that can occur in N'.

Now suppose a third nc-net, N'', is formed from N by adding to N any amount of component redundancy in any manner to correct (or detect) possible jf's, $j > 0$, such that 1) the total cost of the addition does not exceed a(k), and 2) N'' is not one of the a(k), kc (or kd) derivatives of N alluded to above. Let m'' denote the maximum degree of failure which can occur in N''.

Now, the conditions that are asked for in the second question (regardless of whether failure correction or detection is the criterion) are those which insure that the P of N', $P_{N'}$, is always less than or equal to the P of N'', $P_{N''}$. The conditions insuring that the inequality $P_{N'} \leqslant P_{N''}$ holds in the general case are impossible to calculate. However a very loose set of bounds on these conditions can be found as follows: Assume that m' and m'' $<< e^{-1}$. Then since $P_{N'}$ is less than or equal to the probability that any $(k+i)f$, $i > 0$, will occur in N', and $P_{N''}$ is greater than $1/2^n$ times the probability that a single kf will occur in N'', the inequality $P_{N'} \leqslant P_{N''}$ can be approximated by

$$C^{m'}_{k+1} e^{k+1} < e^{k} 2^{n} \tag{1.5.6}$$

which reduces to

$$C^{m'}_{k+1} < e^{-1} 2^{n} . \tag{1.5.7}$$

Thus, a very loose set of bounds are given in partial answer to question 2) above by

Result II: To within a very good approximation, continuously increasing k in most economically deriving kc (or kd) nets is the same as continuously decreasing P in most economically deriving nets with lower P values at least if m' and m'' $\ll e^{-1}$ and $C^{m'}_{k+1} < e^{-1}2^n$ are true.

A little more discussion on the very pessimistic bounds in Results I and II is quite in order. First of all, consider the following example as an illustration of how small e might be in practice. Suppose a single component is located in a computer net which operates at the rate of 10^{-6} seconds between discrete time instants (that is, suppose a one megacycle computer). If this component is to be expected to operate correctly over a period of about 100 hours with a probability greater than 0.9, its e must be less than 10^{-12}. Since general purpose computers are usually composed of upwards of about 10^3 components, each of which is fairly critical to the operation of the over-all machine, a practical upper bound on e might be something like 10^{-14} or 10^{-15}.

Hence, suppose that e must be less than 10^{-15} and n (that is, the average n for the various nets in the over-all computer) less than 11. Then some pairs of m' and k which satisfy the requirements in Result I are given in Table I.

Table 1 - Variance of m' with k

k	m'
1	$(1.4)(10^6)$
2	$(1.8)(10^4)$
3	$(2.2)(10^3)$

Now a final remark regarding Result II: It is obvious that on the average it will always cost less to most economically add enough redundancy to a net to enable it to correct a kf than to enable it to correct a (k+i)f, $k, i > 0$. Thus, in almost all cases, the approximating formula that was used in equation (1.5.6) to replace the inequality $P_{N'} \lessgtr P_{N''}$, is outrageously pessimistic. This together with Table I tend to justify the goodness of the kc and kd criteria for nc-nets.

Note that the possible periodicity of outputs of c-nets make it virtually impossible to frame any manageable "degree of failure" index corresponding to the kf for nc-nets. Hence the nature of 2) and 3) in the expressed objective of this study seems appropriate.

1.6 INTRODUCTION TO THE SOLUTION

This Section discusses some negative conclusions concerning the solution of the kc and kd problem. Its purpose is to point out some prominent defects of a few suggestions which have appeared from time to time in the literature, at least insofar as these suggestions apply to the present study.

One suggestion that continues to appear is to put several computers alongside each other and operate them in parallel. The intent is usually to achieve a unit which is approximately kc. The method almost always involves a comparison of the outputs of 2k+1 computers and a subsequent selection of the majority output as the output of the unit. The biggest shortcoming of any such scheme is that the final output is only as reliable as the selection and final output equipment. In particular, if some subset of the set of components just preceding the final output fail, just as many of the final outputs will be incorrect. In other words, no single output state can be any more reliable than the single component which produces it.

Recently it has also become popular to suppose that satisfactory solutions to the k-degree failure correcting and detecting problem exist in the domain of coding theory. It must be remembered, however, that one of the first premises of coding theory is that the terminals of a channel are perfectly reliable. Since presumably there does not exist a physical component which is perfectly reliable, classical coding theory as it stands does not strictly apply to the kc and kd problems. Furthermore, the terminal trouble cannot be circumvented by translating all the failure's probabilities that are associated with channel termini into the error structure of the channels themselves. For if it could, a method would have to exist for designing perfectly reliable encoders and decoders from unreliable components, which is impossible. Consequently the solution for this study is not strictly contained in the domain of classical coding theory.

Finally, it is apparent that if reliable computers (as they are presently organized) are to be realized from unreliable components, the logic in them must be deterministic in the large and statistical in the small. Here "large" and "small" apply to time, space, or a mixture of both forms of redundancy.

CHAPTER 2

kc NETS

2.1 INTRODUCTION

The purpose of this Chapter is to define a form for kc nets which is probably most economical. This form is specified in terms of the corresponding irredundant net. It is shown that it will always give the correct outputs even if it contains k permanently bad components. It is also shown that the use of a certain type of time-domain redundancy is very undesirable as a means of effecting kc nets.

2.2 MINIMUM OUTPUT REPRESENTATION

At the end of Ch apter 1 it was pointed out that kc nets, $k>0$, cannot have only one wire per binary output channel. Here, a binary output channel of a net, N, is a set of i wires leading from N, $i>0$, over which information is transmitted at the rate of 1 bit every specified number of time instants. Define an individual output from a net, N, to be 1 bit of information from N over a binary output channel. A nc-net, N, is said to realize a logical function $f(x_1, \cdots, x_n)$ if N puts out 1 information bit, corresponding to each of N's complete input sets, $x_1, \cdots, x_n$, telling the value of $f(x_1, \cdots, x_n)$according to a predetermined representation of the binary values of f, $x_1, \cdots$, and x_n. A complete input set to N consists of all the variously timed information-bit inputs to N which act as variables in the production of 1 information-bit output of N. Now this Section proceeds from the 1-wire- output result of Chapter 1 with

LEMMA I: Suppose a binary-output-channel, nc-net, N, which realizes some logical function, f, is given, and that kf's are permitted within N. Then N's output channel must have at least 2k+1 wires if N is kc and if it is ruled that each of N's individual outputs must be observed at only one time instant.

PROOF: Suppose N's output channel has m wires. Since the rule for determining the value of the individual outputs over this channel must be effective[1], the 2^m possible channel state configurations must be dichotomized. One of the resulting pair of subsets of configurations must represent 0 and the other 1. If not, some output configurations are equivocally 1 and 0. Since with each output any k of the m output wires can be in the incorrect state, the subset of configurations representing 1 must contain all of the 2^m members of the set which are distant[2] less than k+1 from the 0f representation of 1. Similarly, with the 0-output subset. This requires $m > 2k$, which proves the Lemma.

2.3 THE 2k+1 FORM

The purpose of this Section is to define a form for economical binary-output-channel, kc, nc-nets which must have their individual outputs observed at only one time instant. Throughout the remainder of the Section, all discussionswill be restricted to binary-output-channel, nc-nets which have their individual outputs observed at only one time instant.

Generally speaking, the nets that are developed in this study are most applicable to the computer field. There, the outputs of nets realizing separate logical functions either have to be used as inputs to other similar nets, or are

1. A fixed procedure which is invariant over all its variable values.

2. The distance between any 2 elements of the set of configurations is the number of places in which the 2 configurations differ.

disposed of so that their effects can recur as inputs to the nets which produced them. Thus, it shall be required in this Chapter that every binary output channel from a kc net match each of its corresponding binary input channels in terms of the number of wires in the channel, and the interpretation and poorest permissible quality of the signal. The quality of an input or output of a net refers to the number of places in which the corresponding signal is corrupted by the effects of failure. Thus, the minimal input-output form for a kc net N(f) which realizes a logical function f, in accordance with the restrictions imposed in this section, is shown in Fig. 6.

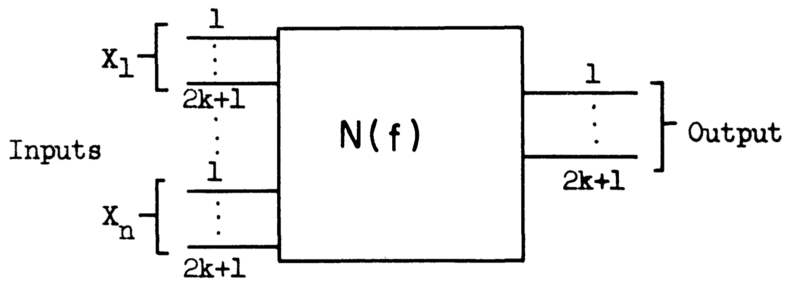

Figure 6. Minimal Input-Output form for a Realization of f.

There, each of N(f)'s inputs, $x_1, \ldots, x_n$, and its output must occur over at least a (2k+1)-wire bundle. The correct state of each bundle at any time instant is taken as the majority state among the 2k+1 individual wire states in the bundle. Thus, a perfect bundle-state configuration consists of all 1's for a bundle state of 1, and all 0's for a bundle state of 0.

Now a form for N(f) which is kc will be given. After it is explained, its optimality properties will be stated and proved.

The form is shown in Fig. 7. In order to facilitate its explanation, first of all assume that all its components are perfectly reliable. Then after the explanation using this assumption is completed, the components' unreliabilities will be recognized and the effects observed.

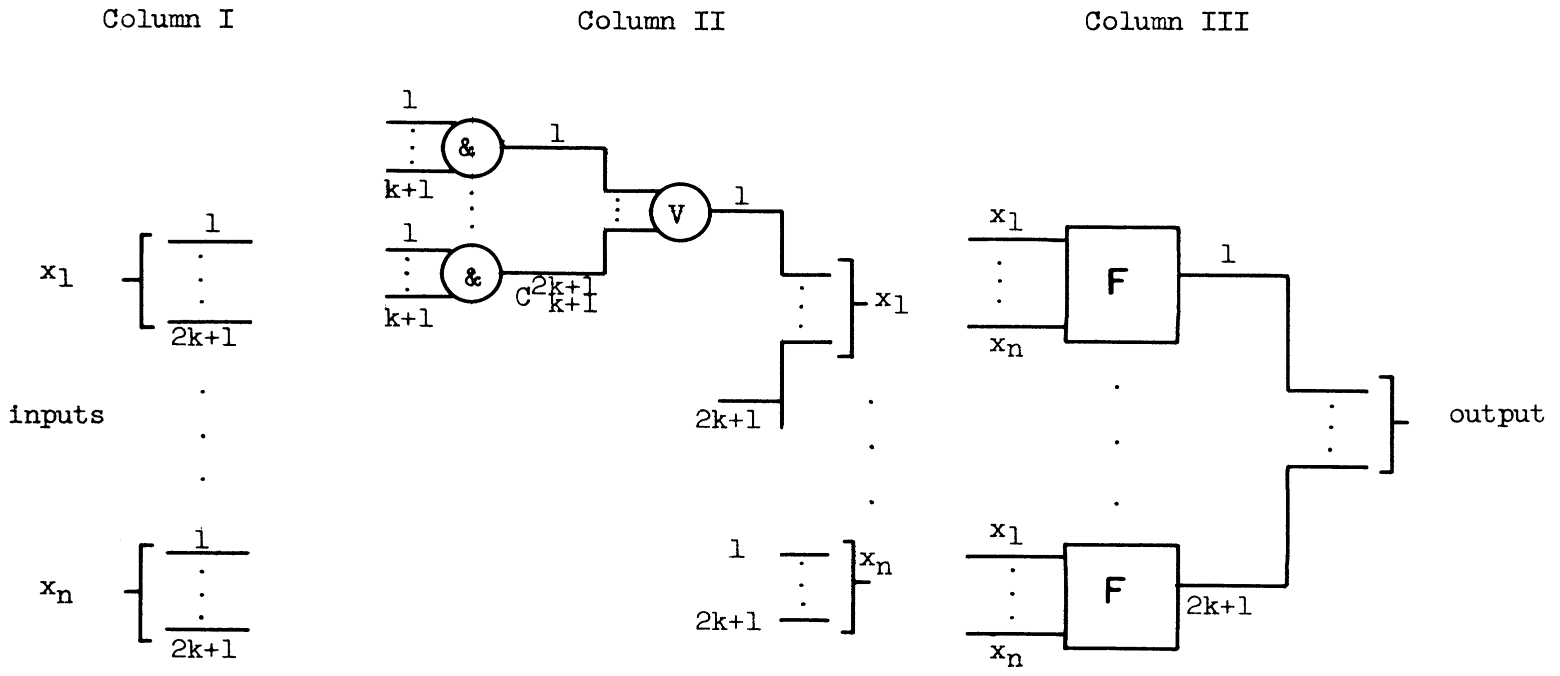

Figure 7. Schematic Representation of N(f).

Column I shows the inputs to N(f). In column II, the corresponding input states are resolved and regenerated. This is done for each input variable as follows: Each (k+1)-wire subset of the (2k+1)-wire input bundle is fed into an "and" component. Then the C^{2k+1}_{k+1} resulting "and" outputs are fed into an "or" component. So the output of the "or" is 1 if and only if the input bundle state is 1. 2k more copies of the resolver net just described are added to produce a 2k+1 wire bundle, each wire of which is 1 at t + 2 if and only if the state of the corresponding input bundle is 1 at t. Such a resolver is indicated schematically for x_1 in Fig. 7. Column II in the Figure is meant to contain a (2k+1)-wire-output resolver net for every one of the input variables.

Now suppose the function which N(f) realizes is irredundantly realized with delay d by the F-net shown in Fig. 8. Then 2k+1 of these F-nets are used

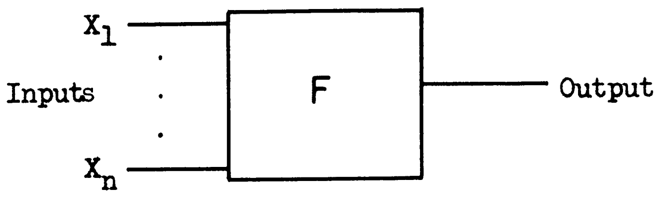

Figure 8. An Irredundant Realization of f.

in column III of Fig. 7 as shown. The ith wire, i=1, ..., 2k+1, of each regenerated input bundle, $x_1, \ldots, x_n$, shown at the right in column II is fed into the ith F-net in column III. The output of N(f) consists of the outputs of the 2k+1 F-nets. Each wire in this output is 1 at t+2+d if and only if $f(x_1, \ldots, x_n)$ is 1 for the $x_1, \ldots, x_n$ bundle state values sent into N(f) at t.

Now recognize the component unreliabilities in N(f) by allowing kf's in the net. It is immediate that if a kf occurs with respect to t in N(f), or in fact if N(f) contains any set of k or fewer bad components, at least a majority of the output wires of N(f) will have the correct state at t+2+d. For the nets producing each of the 2k+1 output wire states of N(f) are disjoint. For example the net which produces the ith output of N(f) is shown in Fig. 9. The wires marked "x_{j_i}" in the Figure are the ith wires of the regenerated x_j input bundle. F_i is the ith F-net, and f_i the ith output wire of N(f).

As for the cost of N(f), it needn't be as great as the form in Fig. 7 shows. For, the resolving function realized in column II of the Figure can be realized much more economically than is shown there. This can be done by using many levels of "and" components instead of just one. Hereafter, a 2k+1 form(or 2k+1 derivative of N) refers to a net which has the form shown in Fig. 7 except that its resolving function is realized most economically instead of as shown there.

2.4 THE OPTIMALITY PROPERTIES OF THE 2k+1 FORM FOR BINARY-OUTPUT-CHANNEL kc NETS

A Lemma will now be stated as a first step in deriving the optimality properties of the 2k+1 form defined in 2.3.

LEMMA II: Given a 2k+1 wire binary-output-channel, kc, nc-net, N, which must have its individual outputs observed at only one time instant and which realizes the logical function f from perfectly reliable inputs[1], N must cost at least as much as 2k+1 separate, irredundant realizations of f.

PROOF: Number N's 2k+1 output wires from 1 to 2k+1 in some arbitrary order. Then let Ni and Nj be the subnets of N which respectively contain all the

1. Note that this is an exception to the requirement placed early in Section 2.3 that all kc nets' outputs match their corresponding inputs.

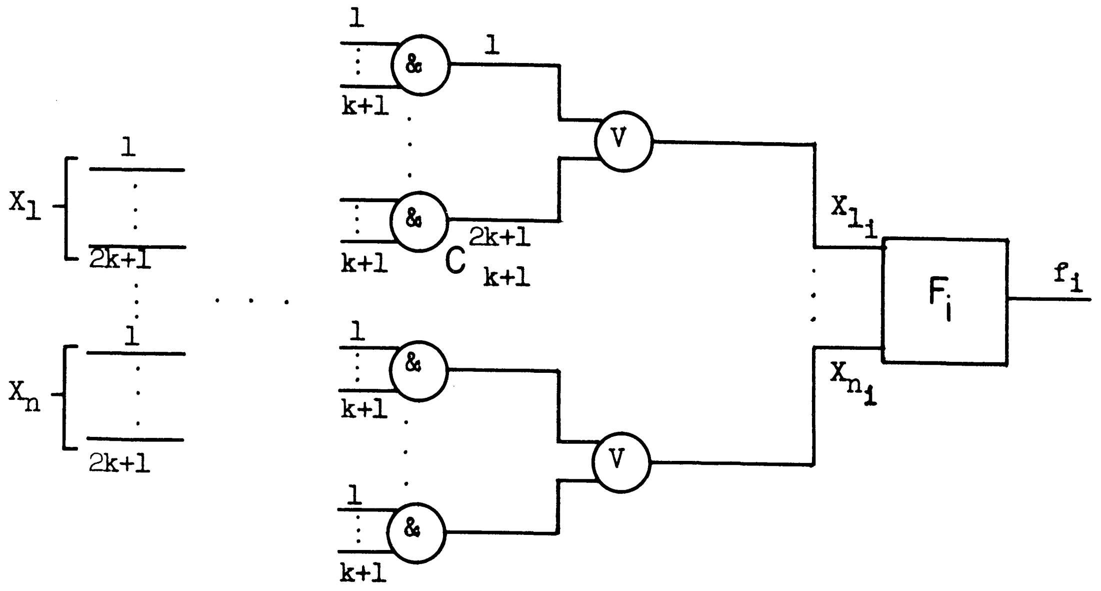

Figure 9. The Net Which Produces the ith Output of N(f).

components in N whose outputs can possibly ever influence (for at least one complete input set) the state of N's ith and jth output wires. Now suppose Ni and Nj overlap: that is, suppose they contain a set, C, of components which are common to both Ni and Nj. Then since N is nc, at least one of these components, say cl, must produce outputs which cannot later affect the outputs of any of the other components in C. Remove cl from both Ni and Nj. Also remove all the rest of the components in Ni and Nj respectively that are thereby cut off from the outputs of the residual Ni and Nj subnets. (obviously cl could not have originally produced Ni's or Nj's outputs). Now denote the output variable which cl produced, y. Then if Ni's and Nj's complete input sets are labeled $x_1, \cdots, x_n$, the outputs of the residual Ni and Nj subnets are value representations of $f_i(y, x_1, \cdots, x_n)$ and $f_j(y, x_1, \cdots, x_n)$ respectively. By a well known theorem of the propositional calculus, f_i can be written as $y \cdot g_{i1}(x_1, \cdots, x_n) v y' \cdot g_{i2}(x_1, \cdots, x_n)$ and f_j as $y \cdot g_{j1}(x_1, \cdots x_n) v y' \cdot g_{j2}(x_1, \cdots, x_n)$

Since both f_i and f_j must equal f, $g_{i1} = g_{j1}$ and $g_{i2} = g_{j2}$. Also, by hypothesis $g_{i1} \neq g_{i2}$ or y could not possibly affect Ni's output. Hence for some complete input set to N for which $g_{i1} \neq g_{i2}$, Ni's and Nj's outputs both depend upon the value of y. For this input to N, if only cl fails, two of N's output wires are put in the wrong state. Hence a kf could cause N to give an incorrect output. But this contradicts a premise of the Lemma. So Ni and Nj must be disjoint for all $i \neq j$. The Lemma follows immediately from this.

From Lemma II, the next step is

<u>LEMMA III:</u> Given any irredundant binary-output-channel nc-net, N, which realizes the logical function f, its 2k+1 derivative, N, is a most economical (2k+1)-wire output channel kc realization of f, at least if it is required that all of N's individual outputs be observed at only one time instant.

<u>PROOF:</u> By Lemma II a 2k+1 wire output kc net which realizes f from N's input variables must cost at least 2k+1 times as much as N. Now if N's kc derivative is to have its outputs match its inputs, each input variable must be presented to the over-all net over at least 2k+1 wires. This means that the inputs to the derivative must have their states resolved in a set of resolving nets. This resolving function, and N's function, are entirely independent both from the logical and cost-of-realization points of view (the last stage of any resolver realization here must be an "or", and all previous stages "and's"). Thus the resolving nets are entirely separate parts of the over-all derivative.

Now each input must be resolved at least 2k+1 times or it can be incorrectly interpreted, as a result of a kf, to the part of the over-all net which realizes f from the resolved input variables. Hence the Lemma by the definition of a 2k+1 form.

Now suppose that N in Lemma II is not required to have only 2k+1 wires in its binary output channel. Does the assertion of the Lemma still hold? At present the author has not yet been able to prove either that it does or does not. It seems from the information which is available in the statement of the Lemma that the only proof that could ever be constructed along this line is a counter example. The reason is as follows: Let N of Lemma II have q, $q >$ 2k+1, instead of 2k+1 wires in its binary output channel. Then it is easy to show that k components can be selected in N such that a majority of N's output wires can each have their states dependent upon one of these component's outputs for some input to N. The trouble is that it cannot be proved that a majority of N's output wires can <u>all</u> have their states dependent upon the k components' outputs for the <u>same</u> input to N. This fact has caused all the presently tried arguments for a proof of the generalized Lemma II to break down. (As suggestive as this might seem for the construction of a counter example, it has not lent any real support in this direction as yet.)

One important fact relating to the situation above is that as q increases, the cost of resolving grows rapidly. In fact, the resolving function must be an "or" function of $\sum_{j=1}^{k} C_j^q$ q-variable "and" functions. So as q grows larger than 2k+1, the cost of realizing the resolving function goes up roughly combinatorially. With this in mind, the following strong conjecture is made.

<u>CONJECTURE I:</u> Lemma III holds even if the condition that the kc derivative have only 2k+1 output wires is removed. <u>(Hereafter whenever it is desired to premise the truth of Conjecture I in the statement of a theorem, lemma, result, or conjecture, and asterisk (*) will be placed after the word "theorem", "lemma", "result", or conjecture" respectively.</u>

Now let us consider the possibility of reducing the cost of deriving the kc form of N in Lemma III by substituting time-domain redundancy for equipment-domain redundancy. Since there can be no feedback loops in a nc-net, time-domain redundancy can be utilized at this point in only one way. That is by letting each state quantity which is involved be represented by an ordered sequence of 0's and 1's instead of just a single 0 or 1. The reason why in general this cannot lead to a satisfactory solution to the kc problem is illustrated by the following heuristic example: Suppose three different gnc-nets are given, and that each 0 or 1 state quantity involved in their operation is represented by a sequence of 2k+1 ordered 0's and 1's. Let the state represented by each of these sequences be the same as the majority state (0 or 1) of the sequence. Denote the 3 gnc-nets S_1, S_2, and S_3 respectively, and let them be as shown in Fig. 10. Allow kf's to occur in each of S_1, S_2, and S_3. Then consider an output

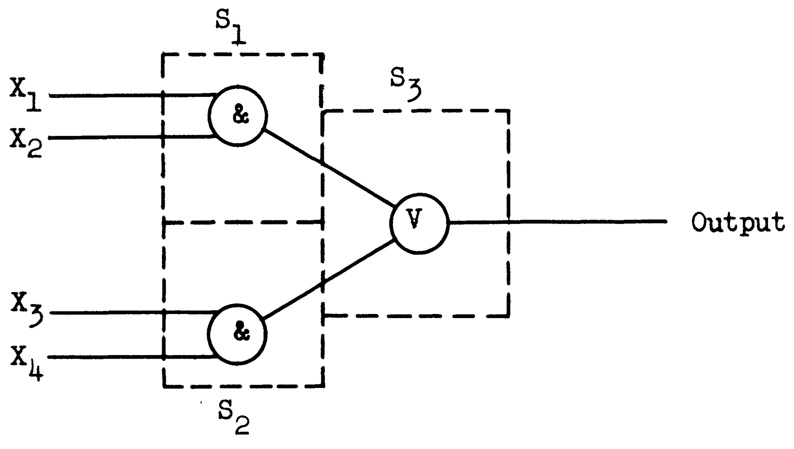

Figure 10. Three gnc-nets.

sequence that might correspond to the following quadruple of input sequences, each of length 2k+1, which start at t: $x_1 = 1, \ldots, 1$; $x_2 = 1, \ldots, 1$; $x_3 = 1, \ldots, 1$; and $x_4 = 1, \ldots, 1$. The outputs of S_1 and S_2 starting at t+1 might be k+1, 1's followed by k, 0's. S_3 might fail from t+2 to t+k+2, so its outputs might be k, 0's, followed by 1, 1, followed by another k, 0's. Thus S_3's output could be put badly in error by failures occurring at different stages in the over-all gnc-net at noncorresponding time instants.

This example heuristically illustrates why time and equipment domain redundancies cannot in general be interchanged in nc-nets which are to be used in over-all computer nets. Time expansions of any length do not insure that output sequences from a nc-net will be of as high a quality as the corresponding input sequences. Thus, there follows

LEMMA IV**: (In the notation of Lemma III) Given N, N is a most economical kc realization of f, provided the quality of such a realization's individual outputs is required to be at least as good as that required of any of the realization's inputs.

The last part of this Section concerns a question regarding the general specification of the F subnet in the 2k+1 form of failure correcting net. The question is: given a large irredundant, binary-output-channel nc-net, N, which realizes the logical function f, how should N be subdivided into F subnets in order that the over-all 2k+1 type failure correcting derivative will be the most economical one[1] realizing a certain P? Hereafter, use failure-correcting derivative of an irredundant net, N, (which includes kc derivative, etc.) only to refer to a net which has the failure-correcting properties specified, and which realizes the same logical functions and/or memory functions as N.

This question can be fairly satisfactorily answered in principle if it is assumed that each subdivision of N is equally critical as far as the correct operation of N is concerned. Begin by letting N_1 be the 2k+1 derivative of N such that N_1 is kc and N_1's F subnets are identical to N. Let m_1 denote the maximum degree of failure in N_1. Then let N_2 be specified as follows: partition

** and 1. Assuming Conjecture I to be true.

N in some arbitrary way into S disjoint subnets $F_1, F_2, \ldots, F_s$, of as nearly equal size as possible. Then form 2q+1 type qc derivatives of each F_i for some q in the range $k/s \leqslant q < k$. Finally, interconnect these derivatives into an over-all net, N_2, such that N_2's output function is the same as N_1's. Denote by m_2 the maximum degree of failure in N_2.

Now the question above becomes: is the P of N_1, P_{N_1}, times the cost of N_1, C, generally less than the P of N_2, P_{N_2}, times the cost of N_2, C_2? Let e be small and approximate P_{N_1} by $C_{k+1}^{m_1} e^{k+1}$. Also, if $P_{N_2}^{i}$ denotes the P of the qc derivative of F_i in N_2, approximate P_{N_2} by $1 - \prod_{i=1}^{s}(1-P_{N_2}^{i})$, or in other words, $sC_{q+1}^{m_2} e^{q+1}$. Let cs be the cost of N. Then approximate C_1 by $(2k+1)(\ (k+2)\ (C_{k+1}^{2k+1})n + cs)$ and C_2 by $(2q+1)\ (\ (q+2)\ (C_{q+1}^{2q+1})n' + c)s$, $n' < n$.

Then the above question becomes, does

$$(2.4.1)\quad P_{N_1} C_1 = (C_{k+1}^{m_1} e^{k-q})\ (2k+1)\ ((k+2)(C_{k+1}^{2k+1})n + cs) < P_{N_2} C_2 = sC_{q+1}^{m_2}$$

$$((2q+1)((q+2)(C_{q+1}^{2q+1})\ n'+c)s$$

hold? Quite apparently, for $e < 10^{-12}$, which was seen in Chapter 1 to be a good practical assumption, this equation generally holds. Thus,

<u>RESULT III:</u> Given a large, irredundant, binary-output-channel nc-net, N, which realizes the logical function f, it is generally best in practical cases to derive N's kc derivative, N, by letting N's F subnets be identical copies of N, rather than first subdividing N and deriving N's failure correcting derivative by interconnecting qc derivatives, $k/s \leqslant q < k$, of each subdivision of N.

---

1. It will be seen below that the following approximations do not materially affect the essential point of the analysis.

Note that Result III, when taken in the light of Lemma IV, amounts to a resubstantiation of the practical equivalence of the kc and P criteria that was asserted in Chapter 1.

2.5 THE OPTIMALITY PROPERTIES OF THE 2k+1 FORM FOR MULTI-OUTPUT-CHANNEL kc NETS.

The purpose of this Section is to extend Lemma IV to pertain to multi-output-channel as well as binary-output-channel kc nets. A multi-output channel net is one which has a set of i wires leading from it, $i > 0$, over which information can be transmitted at the rate of m bits, $m > 1$, each time instant. (From here on, regard an individual output of a net N as 1 bit of information from N.) The first step is to recognize that, for general $r > 1$, and allowing kf's, it does not take at least $r(2k+1)$ wires in parallel to represent r bits of information coming out of a net at one time. This fact is immediate to coding theorists. Hence an example to illustrate it will suffice here. If k and r are equal to 2, $r(2k+1) = 10$. But the following 8-wire code has 4 points each of which is distant[1] at least 2k+1, or 5, from the other 3 points, making the code kc[2]: (11111111), (11000000), (00111000), (00000111). The extension of the argument for kc codes to kc nets is obvious.

One point of this Section is that, while it is possible to use fewer than (2k+1) wires to represent an instantaneous r-bit output of a net, it is probably uneconomical to do so. This is best explained by presenting the general argument from an example.

Let N be a nc-net which is presented for reliability improvement, where N has two inputs, x_1 and x_2, two outputs, y_1 and y_2, and cost c. Then the

1. See the footnote in Section 2.2 for a definition of distance.
2. That is, k-error correcting in the usual coding sense.

2k+1 derivative of N, for k=1, is shown in Fig. 11. Its cost is 54+3c. Note that as long as each individual output of an over-all kc net is produced over a binary channel, each of the net's output channels must contain at least 2k+1

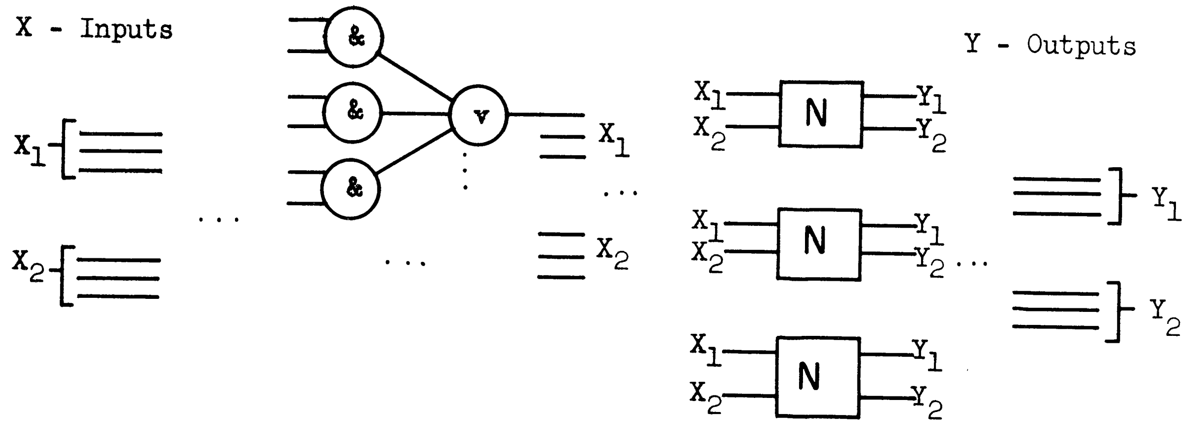

Figure 11. A 2k+1 derivative, k=1, of N.

wires. Note also that in this case, by Lemmas* I to IV, each individual output must be derived in a set of subnets that costs at least as much as 2k+1 separate irredundant realizations of the same output function and the corresponding 2k+1 type input resolvers (it is assumed here that the poorest permissible input and output signal qualities are the same). Hence

THEOREM* I: Given any irredundant nc-net, N, which realizes the set of m logical functions, $\mathfrak{F}$, m≥1, its 2k+1 derivative is a most economical kc realization of $\mathfrak{F}$ if it is required that this derivative produce each of its individual outputs over binary channels. In fact this derivative can correct the effects of any set of k or fewer permanent component failures which might occur within it. These assertions are true provided it is assumed that all the input and output channels of the derivative must match in terms of the number of wires in the channels, and the interpretation and required qualities of the signals they carry.

* Assuming Conjecture I to be true.

Now, to continue the example begun above, suppose N's kc derivative need not produce each of its individual outputs over binary channels. Then if it is desired, the 6-wire x-inputs and y-outputs of N's 1c derivative can be reduced to 5-wire bundles by using the 1c input- output code shown in Table 2. Using this code, the 2^5 possible state configurations of the input and output bundles must be partitioned into 4 disjoint sets, where every member of each set representssome configuration of x_1, x_2, or y_1, y_2. Thus, the 32 possible 5-wire state configurations can be partitioned into 4 sets of 8 members each as shown in Table 3. There, each of the 4 groups of code points contains at

Table 2. A 1c code

x_1, x_2 or y_1, y_2	5-wire code points
11	11111
10	00010
01	11000
00	00101

Table 3. A 5-wire output code

The leading set of each column is the perfect code point.

5-wire code points \ x_1, x_2 or y_1, y_2	1 1	1 0	0 1	0 0
	11111	00010	11000	00101
	11110	00011	11001	00100
	11101	00000	11010	00111
	11011	00110	11100	00001
	10111	01010	10000	01101
	01111	10010	01000	10101
	10011	01011	01001	01110
	10001	10110	01100	10100

least all points distant 1 from the perfect code point of the group. Now any 5-wire input-output, 1c derivative of N, N# say, must contain the following: 5 realizations of 5 separate functions of the (5) input variables to N#. Each of these realizations must map the input sets for which its corresponding output is to be 1 onto 1, and the remaining input sets onto 0. The realizations may not necessarily be disjoint, but no 1f can be allowed to cause two of their outputs to be incorrect.

Upon reviewing Table 3, one is struck with the realization that in general, at least,a majority resolver can always be much less costly to realize than any of the non-threshold (threshold being p out of q) types of resolvers. The reason is, first, that if kc input-output coding is used in an n-variable input kc net, at least q complete input resolutions are required, where q is the minimum number of places in an n-information-place, kc code. (Of course if kc input-output coding is not used and all input and output channels are binary, threshold type resolving must be used and each input variable must be resolved at least 2k+1 times.) These resolutions must be such that no kf is able to cause more than any k of the resolutions to be incorrect. For otherwise a kf might cause a net's input to be misinterpreted and an incorrect output to be produced. Now each single resolution here must be the realization of an "or" function of $\sum_{j=1}^{k} C_j^q$ q-input variable "and" functions. Thus, on the basis of the known simplicity of the majority type resolution for 2k+1 wire binary channels, the following conjecture is made:

<u>CONJECTURE II:</u> Theorem I holds with the condition that "it is required that this derivative produce each of its individual outputs over binary channels" removed. <u>(Hereafter whenever it is desired to premise the truth of Conjecture II in the statement of a theorem, lemma, result, or conjecture, a double asterisk (**) will be placed after the word "theorem", "lemma", "result", or "conjecture" respectively.</u>

CHAPTER 3

MEMORY NETS

3.1 INTRODUCTION:

This Chapter presents the solution to the second problem of the study: namely, the problem of defining a most economical form of memory net which is compatible in some failure-correcting sense with the kc nets of Chapter 2. Since memory nets must be cyclic, the kf and kc notions cannot strictly apply here until all closed loops are cut. Using this approach, a form of memory net is developed which is probably most economical. The development includes: the listing of some independent necessity conditions; the corresponding specification of a most economical net for storing at least 1 bit; and finally the specification of an economical form of multi-bit memory net.

3.2 SOME NECESSITY CONDITIONS AND THE 1-BIT MEMORY NET

It will hereafter be assumed that whatever type of memory net is developed, it will have to be controlled through components which can fail both transiently and permanently. Thus, the effects of different transient failures in these controlling components could accumulate in the memory nets, if permitted to, and eventually produce an incorrect memory state. Consequently, if reliable long-term storage service is to be obtained from a net, that net must frequently regenerate its own storage state. It will be able to do this only if it contains a feedback loop.

The type of feedback which is required here is illustrated in the irredundant memory net, N, shown in Fig. 12. This net,(which is <u>not</u> intended to be kc in any approximate sense) operates as follows: a 1 can be stored in N at t by presenting a 1 on at least one of the $S_1, \cdots, S_n$ "set" inputs of N at t, and by having

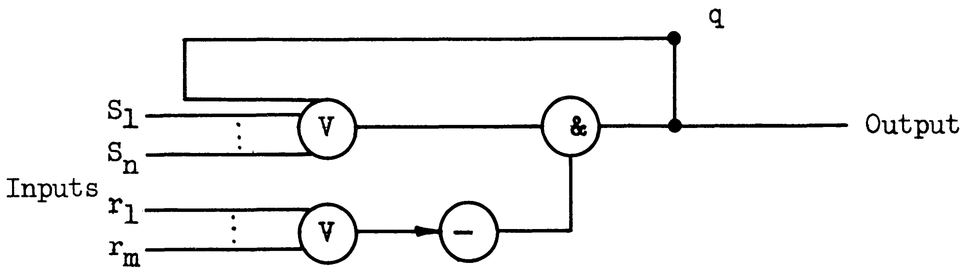

Figure 12. An Irredundant 1-Bit Memory Net.

0's present on all of N's $r_1, \cdots, r_n$ "reset" inputs at t. This 1 then circulates around N's feedback loop until a properly timed 1 on one of the reset inputs of N destroys it. The timing must be such that the "and" component input coming from the reset branch of N is 0 just as the circulating 1 arrives at the other "and" input. Note that even though N is irredundant for a 1 bit capacity, it can actually store 2 bits due to its 2-unit loop delay.

Now a derivative of N will be described which is approximately kc. To develop this derivative, first of all cut the feedback loop in Fig. 12 at any point, "a" say, and thereby obtain a nc-net. This nc-net's most economical kc derivative is the corresponding 2k+1 form, which will be denoted $\bar{N}$ throughout the remainder of the Chapter. A memory element, E, capable of storing at least 1 bit can be made by feeding $\bar{N}$'s output back as a "set" input. The result is shown in Fig. 13 (except that for convenience the input resolving subnet shown there is not most economical). In the Figure the "v_1" and "v_2" components perform the functions of both the normal resolver stage "or's", and the input "or's" of E's logical subnets. Here each of E's logical subnets is of the form shown in Fig. 12 except that it does not have a feedback loop as shown there. Note that for generality in Fig. 13, none of E's

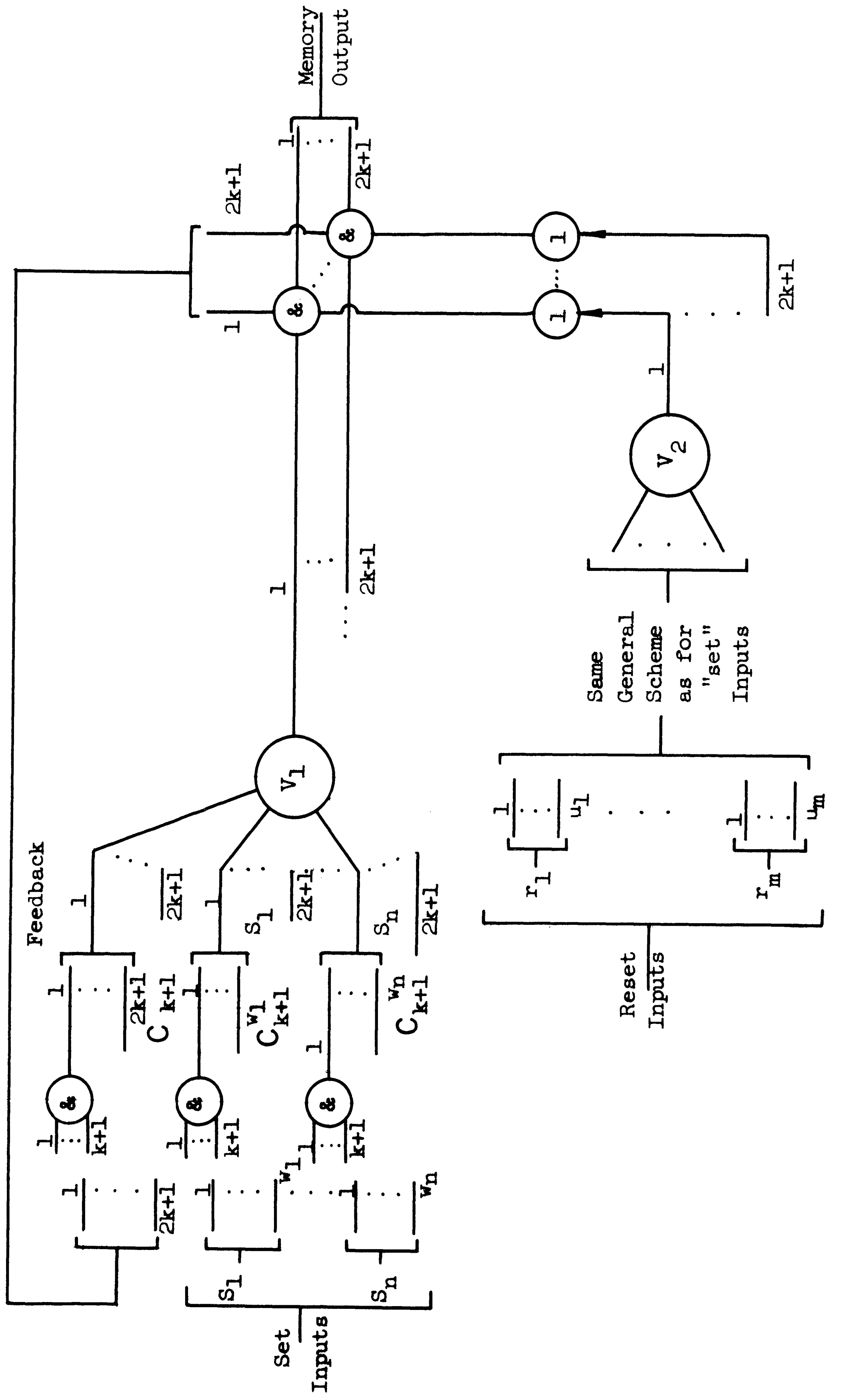

Figure 13. A 2k+1 Memory Net.

input-variable bundle sizes is shown as 2k+1. The only restriction placed on these sizes is that they all be at least as great as k+1.

E's properties are as follows: E has a loop delay of $d > 1$ units, enabling it to store d bits of information. Hence E must be used on a modulo d time scale, with corresponding set and reset input times coinciding. The nc-net, $\overline{N}$, from E's inputs to E's output is kc. In fact this net can correct the effects of any set of k or fewer permanent component failures which might occur within it. E can be set or reset by the 2k+1 type kc nets of Chapter 2. In this case $w_1, \cdots, w_n, u_1, \cdots, u_m$ are all 2k+1. Finally, <u>E is the most economical form of net that can reliably and indefinitely store at least 1 bit while: 1) allowing kf's to occur within any of its unclosed loops; and 2) allowing its set and reset signals to come through nets where transient type kf's are permitted.</u> With regard to 1) above, in general in this study, an <u>unclosed loop</u> in a net will refer to any set of components in that net which were previously interconnected so as to include one or more feedback loops, but which since have been disconnected in a particular way at least enough to break up all these loops.

The assertion containing 1) and 2) above requires a brief argument. It is immediate that if a stored bit is represented by a parallel configuration in a channel, at least 2k+1 wires are necessary in that channel. For no net with fewer than 2k+1 wires in its storage channel can guarantee meeting its input quality requirements at its output. And this is necessary by the part of this Section on state-regeneration requirements. Since it is assumed that the w_i and u_i are not variables here, the truth of Conjecture I for this case is obvious and the assertion is proved.

3.3 MULTI-BIT MEMORY NETS AND THE FINAL THEOREM.

It was conjectured in Chapter 2 that although it is possible to use fewer than r(2k+1) wires to represent an instantaneous r-bit output of a net, it is not most economical to do so. The reason lay in the relatively low cost of the majority type of resolving subnet with respect to the costs of other types. The essential points of the example and argument in Chapter 2 apply even more dramatically to multi-bit memory nets. Hence very probably it is not most economical to store r bits in one channel, $r>1$, by making use of corrective coding techniques.

It is apparent from the previous Section then that if Conjecture II is true, the most economical way of storing r+d bits[1] and allowing 1) and 2) at the end of the previous Section, $r \geqslant 0$, is by inserting r "or" components in series in each wire of the feedback loop of the 2k+1 type memory element described in that Section. Let any memory element of this form be denoted an <u>M_r element</u>. Then there follows

<u>THEOREM**II:</u> Given a net, N, which consists of an array of $-[-b/(r+d)]$ disjoint elements[2], each of which is an M_r element, N is the most economical net which can reliably and indefinitely store b bits while: 1) allowing kf's to occur within any of its unclosed loops, 2) allowing its set and reset signals to come through nets where transient type kf's are permitted; and 3) not allowing any bit to be inaccessible at its output for any more than r+d consecutive time instants. The Theorem assumes that: 1) the b memory locations are placed among the M_r's in N to take full advantage of any coincidence which might be present among the locations' "set" and "reset" input variables; and 2) there is no necessity, after every

1. Where "d" is as on the previous page.
** Assuming Conjecture II to be true.
2. "[a]" is the greatest integer less than or equal to a.

rearrangement of locations, of isolating inputs to one memory location from those to another location in the same M_r of N. If there is, enough more elements must be added to N to meet this necessity. In general, this may dictate the use of different r's in the various M_r's in order to minimize the over-all cost.

A memory location in a memory, M, is a time-ordered set of geometric positions in M where 1 bit is kept in storage. Thus, one of the d memory locations in an M_r element, r=0, is at the outputs of the element's "or" components at t, its "set-reset" "and" components at t+1, its first level resolver subnet "and" components at t+2, etc.

CHAPTER 4

OVER-ALL FAILURE-CORRECTING COMPUTER NETS.

4.1 INTRODUCTION

The results that have been obtained thus far are summarized in Theorems I and II and Conjecture II. These results specify definitions for memory and nc-nets which are kc in the senses indicated and which are probably most economical for their failure correcting properties. At this point then, the next thing is to specify how to make over-all failure-correcting computer nets from the types of nets that have been discussed thus far.

In the present Chapter, two such general forms are developed — and both from an outline which is composed of a nc-subnet and a memory subnet. The reason for developing the two forms from this two-subnet outline is twofold: First of all, the outline enables a set of rather strong economy claims to be justified for the two forms on the basis of the results in Chapters 2 and 3. Secondly, the outline furnishes a fairly good analytical model of present-day computers.

In the first form, the two subnets are constructed in accordance with the failure-correcting derivative nets defined by Theorems I and II respectively.

The second form is characterized by having its nc-subnet realized with fewer than 2k+1 disjoint copies of its corresponding irredundant net. In this sense it differs greatly from previously preferred failure correcting nets. Its memory subnet, however, is constructed in accordance with Theorem II.

An alternative variation of the second form is described which has a kd nc-subnet instead of a kc one. The variation is the result of an attempt to shift from pure equipment-domain redundancy to maximum time-domain redundancy. The method in this variation is to arrange for each output of the kd net to be ignored unless it contains purely correct information. The possibility of having k permanent failures in the kd net is allowed for by including reserve nets which are switched into operation from standby locations.

The Chapter shows that in every case, one of the variations of either the first or the second general form referred to above is the most economical k-degree failure-correcting form for an over-all computer net of the two-subnet outline.

4.2 THE TWO-SUBNET OUTLINE

The purpose of this Section is to define the two-subnet computer outline mentioned in Section 4.1. The outline is shown schematically in Fig. 14 below. There, M is the net's memory subnet and NC its complementary nc-subnet. Each of these subnets is constructed in accordance with the equipment allowances made earlier in the study. The directed lines into and out of the M and NC blocks represent information channels, none of which is in general binary. In fact, the channel capacities (in bits) from NC to M and from M to NC are in general equal to twice the number and the number respectively of separate memory elements in M. The 2-to-1 ratio here is due to memory elements having both set and reset inputs.

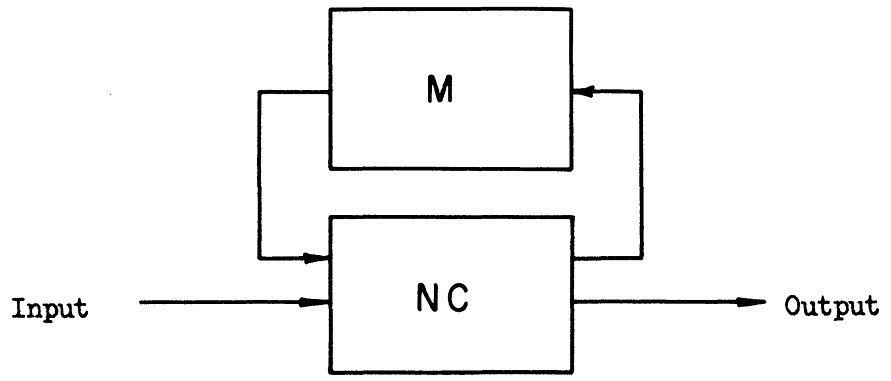

Figure 14. An over-all Computer Representation.

Now designate the whole net in Fig. 14 N^*. Also denote by I_j the jth input to NC; S_j the state of M at the arrival of I_j; and O_j, N^*'s output from NC corresponding to I_j and S_j. Here the state of M refers to the configuration of stored values in the memory locations of M.

Now the sequence of operations in N^* is as follows: Suppose M is in state S_0 and I_0 arrives. Then NC produces O_0 and alters M's state to S_1. Next I_1 arrives, and O_1 is produced and M's state goes to S_2. And so on.

In general, in the over-all sequence of operations in N^*, several of the I_j are required to be vacuous. When a vacuous input, say I_j, is required, the preceding change in M from S_{j-1} to S_j is sufficient to cause the production of O_j and the accompanying change from S_j to S_{j+1}. So in effect, each S_j controls by its state whether or not a next input, I_j, is to be requested or not. A request for another input by M, as indicated by the state of M, corresponds to the carrying out of a "read" instruction in an ordinary computer. It is assumed that NC operates on S_j at each time instant in such a way that all the successive S_j are as desired. This type of operation can be achieved by using ordinary sequential net design principles. In fact, by these same principles, N^* may be designed so that its M block is just a straight connection of wire bundles from NC's output back to its input. Finally, assume N^* is irredundant for a net having its properties; and hereafter say that any net which is constructed in the general form of N^* is of the form of N^*.

Quite apparently N^* is as general a computer representation as any completely finite Turing machine, T (that is,one having a finite tape and finite numbers of internal state and tape symbols). For the position of T's tape, its data, and its state can all be represented in M; and NC can be designed so its operations on M represent the sequence of operations in the machine proper.

4.3 THE OVER-ALL 2k+1 FORM OF COMPUTER.

The purpose of this Section is to describe one variation of a form of over-all failure-correcting computer net. This form, the first of two such to be specified in this Chapter, is characterized by being constructed in accordance with the kc derivatives defined by Theorems I and II. Two variations of it follow in this and the next Section.

The first variation follows quickly from Theorems I and II and the computer outline defined in relation to Fig. 14: This variation is of the form of N^*. Its M net is entirely composed of M_r type memory elements, and its NC net is a 2k+1 type nc-net.

NC's operation into M is so as to produce the proper sequence of states in M. The r's of the M_r's in M and the delays in NC are such that the arrivals of information (from M) at NC and (from NC) at M are timed to permit this proper sequence. The correct timing of information signals into NC and M can be accomplished by using single-input "or" components to produce delays where they are needed. To illustrate how this may be done, two examples follow. In these examples, for simplicity, assume that all I_j are vacuous, and that no O_j are produced until the end of the computation. The generalization to cases where these special conditions do not hold will be immediate after the examples have been presented.

The first example, a net of the type of the first variation defined above, has an r of 0 in all of its M_r type memory elements. Thus, the content of each of its memory locations is available at its M net output every dth time instant.[1] Now in

1. Using the notation of Theorem II.

general, in order for the example's NC net to determine its M subnet's $j+1$ st state from this subnet's jth state, NC must have access to the stored contents of all of M's memory locations. One way to insure this is to store only 1 bit in each of M's memory elements. Then all the stored bits can be synchronized to appear simultaneously at M's output every dth time instant. M's output can be made all 0's at each d-1 successive intervening time instants, bundle-state 1's can be regarded as the only positive outputs from M and NC. (This is convenient, since only 1's can set or reset M_r type memory elements anyway). Finally, NC can be designed so that it will not produce a positive output unless the corresponding input is something other than all 0's. With these provisions, at least the timing from M into NC can be taken care of.

Now in order that NC's outputs can be stored in M with the proper timing, the delay from every input time at NC to each of NC's corresponding individual output times can be made ds for some $s > 0$. With such an arrangement, all of NC's outputs must go into the proper third of the memory locations in M. Any extra delays that are needed at NC's normal output points to effect the ds delay characteristic can be instrumented by series connections of single-input "or" components.

The remainder of the design of this example follows from the principles of sequential net design. For these principles can be used to insure correct operation of the over-all net regardless of different s delay factors which might be associated with the various individual outputs of NC. The I_j inputs can be presented to the over-all net either at each time instant, or at every dth time instant along with the outputs of M. The O_j outputs of the over-all net of course can be produced at any time.

The second example which is given to illustrate a possible timing arrangement for the first variation above has only one M_r element in its M block; so in this case r is maximized for one element. Now as above, in general, NC must have access to the stored contents of all of M's memory locations in order to determine each of the successive outputs from NC to M. In this example there are $r + d$ memory locations, and the contents of each of them is available over M's binary output channel exactly once during any interval of $r + d$ time units duration. All the outputs of M from t to $t + r + d$, for any t, can be presented to NC's computing subnet, NCT, at $t + r + 2d-1$ by using the scheme shown in Fig. 15.

The two delay subnets and the NCT subnet shown in the Figure constitute one of the F subnets of the over-all net's NC part. The memory-output state resolver for the F subnets is in the block adjacent to M as indicated. The computing logic of the over-all net's NC part is contained in the NCT block. The outputs of this NCT block are fed into a delay net, and from there into the set and reset control nets of M. A second similar type delay net is located between NCT's input and the M-output state resolver. The C block will be explained later. Only 1's are regarded as positive outputs from M and NCT.

To return to the operation of this example, the effect of the two delay nets in Fig. 15 is: M's resolved outputs are produced d-1 time instants after the corresponding outputs of M are issued. The first delay net serves to line up each of M's complete set of $r + d$ outputs in time so as to present them simultaneously to NCT. NCT puts out all its outputs s time units after the corresponding inputs. It is required that s equal $j(r+d)-d$, for some $j > 0$. In general, extra "or" components must be used as delay elements in NCT in order to achieve

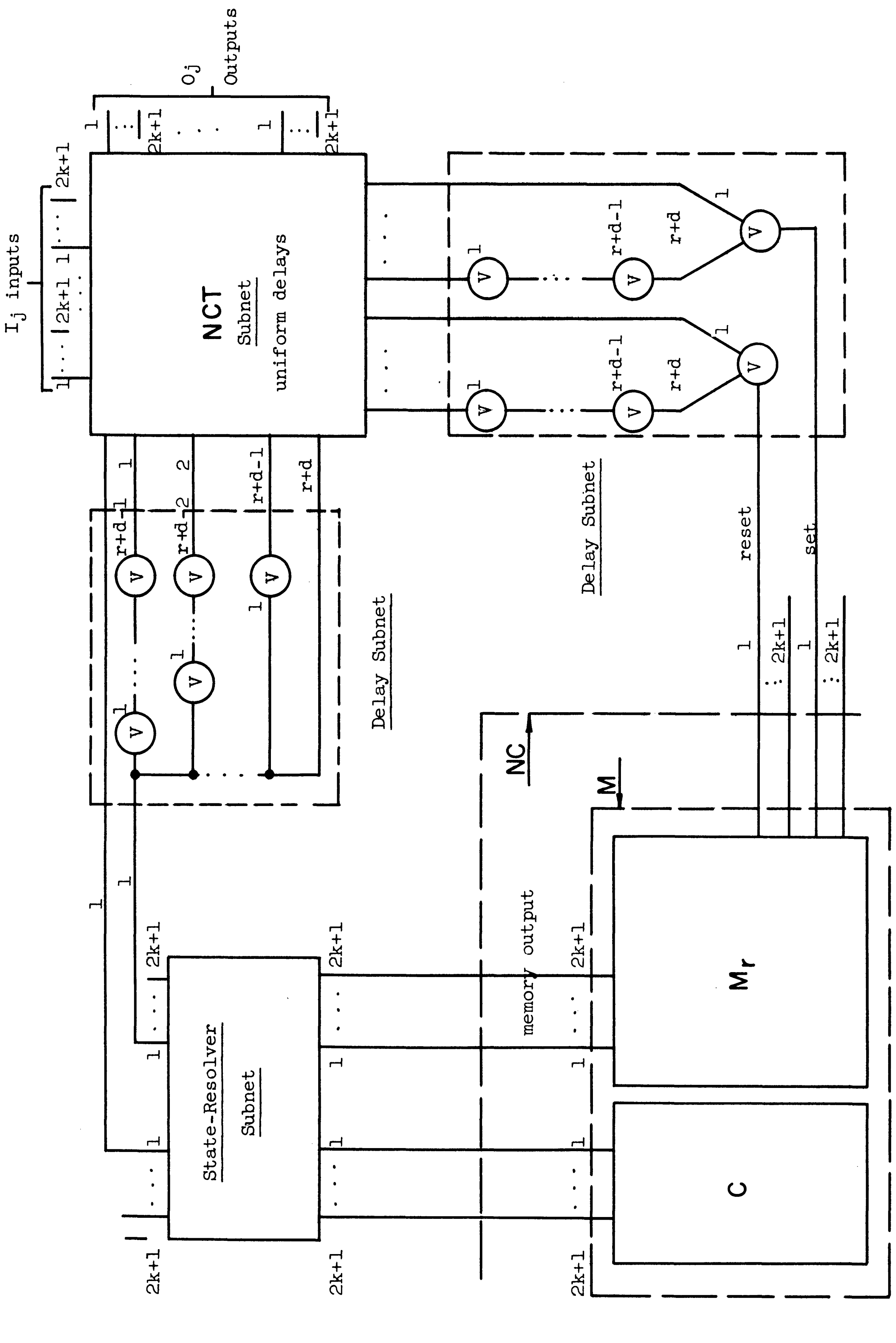

Figure 15. A First Variation Computer Net with r Maximinzed.

this. To continue, NCT's outputs are redistributed in time so as to be stored in the proper locations in M. This is insured since the delay around the loop in Fig. 15 is $(j+2)(r+d)-d$, $j > 0$; and the delay in each of the $2k+1$ "resolver-net-and -component" chains from M's input to M's output is d,

The only thing that prevents the scheme in Fig. 15 from working as has been explained thus far is that M's successive outputs (being the contents of M's successive memory locations) appear on each of the NCT nets' input wires at successive time instants. This must be eliminated by having synchronized clock signals sent into the NCT nets every $(j+2)(r+d)-d$ time instants such that these nets produce positive outputs corresponding only to their inputs at clock times. Such clock signals can be obtained most economically by feeding the outputs of the C net shown in Fig. 16 into the NCT nets. C has only one circulating 1, and this is timed to arrive at the NCT nets' inputs $d-1$ time units after the contents of M's first memory location is put out of M. C is considered as a part of M because of its feedback loop.

Now of course in special cases, it is expected that several of the "or's" in Fig. 15 can be eliminated without affecting the relative timing. But discounting this possibility, the state of M changes only every $(j+2)(r+d)$ time instants.

In Fig. 15 NCT is designed to produce the proper sequence in M, and inputs and outputs are normally fed into and out of NCT. This completes the discussion of the second example.

In general where the maximum r can be very large, there are a wide range of timing arrangements that can be used in the first variation of the over-all $2k+1$ form of computer net being discussed in this Section.[1] However, all of

1 This includes, e. g., those arrangements based on utilizations of additive group generator concepts for additive groups of integers modulo primes.

these ways lead to a form of over-all net which, operationally speaking, can be thought of as follows: It is of the form of N*. All its memory states are available to NC at every instant; so NC alters these states (with delay >0 from NC's input to any of its corresponding outputs) in proper sequence. Now because of the universality of this conceptual model, define CN* to be a net form which is constructed as explained in the second paragraph of this Section and without any particular timing arrangement specified.

At this point the subject of economy is considered. In general the direct effects of the outputs of an N* net's NC net have to be usable later on as inputs to N*'s NC net from N*'s M net. Thus, the 2k+1 form of Theorem** I is the most economical type of kc net that can be used to independently realize the NC part of N*. Theorem** II specifies a most economical type of memory net that can 1) correct any kf's that might ever occur in any of its unclosed loops, and 2) be controlled

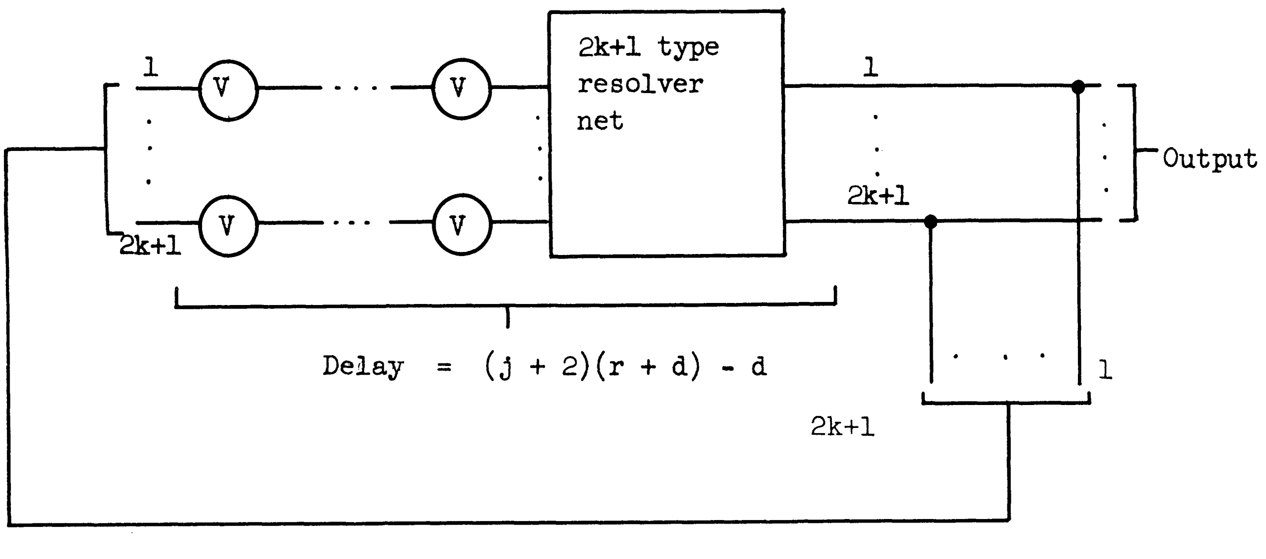

Figure 16. The C Net of Figure 15.

** Assuming Conjecture II to be true.

through nets in which transient type kf's can occur. Hence this type of realization of N*'s M net is the most economical one which has the failure-correcting properties mentioned.

Note that all of the extra costs that are involved in the interconnecting of the realizations of NC and M specified by Theorems I and II arise from one trouble. That is the incompatibility which must always be present between the modes of operation of an nc and memory net. For any time is an appropriate input or output time for a nc-net, but memory locations are accessible only periodically. The important point at this juncture, however, is that the form of M which has been chosen is the most economical one for any possible pattern of storage cycle lengths.[1]

Now Theorem** I refers to independent kc nets, but not necessarily to kc nets which feed into memory nets. Hence, since the cost of a most economical memory net[1] increases rapidly with the size of its set and reset input bundles, any economy claim on CN* has to carry a condition which recognizes this. Note finally that the operation pattern of CN* which has been specified does not use any time-domain redundancy in the inter-operation of its NC and M parts.

The above is the argument for

<u>THEOREM III</u>**: CN* is a most economical form of over-all computer net, 1) which is of the form of N*; 2) which is kc in its NC part; 3) which can a) correct any kf's that might ever occur in any of the unclosed loops in its M part, and b) be controlled through nets in which transient type kf's are allowed to occur; 4) which has NC-to-M communication channels of 2k+1 or more wires per-bit capacity; and 5) which does not use any form of time-domain redundancy.

** and 1. Assuming Conjecture II to be true.

CN* also has the property that it can correct the effects of any set of k or fewer permanent component failures which might occur within it.

4.4 THE SECOND VARIATION OF THE OVER-ALL 2k+ 1 FORM OF COMPUTER NET

The second variation of the first over-all failure-correcting computer form of this Chapter will now be developed. Let there be given any irredundant computer Net, N. Define a cycle of N as follows: a set, A, of components, in N constitutes a cycle of N provided the output of each component in A is fed into the input of exactly one other component in A. An example of a cycle is shown in Fig. 17. In general the cycles of N are not disjoint.

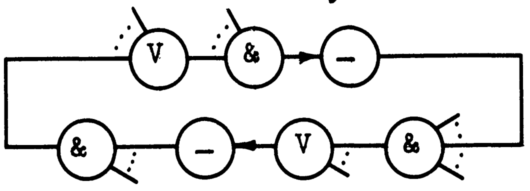

Figure 17. Example of a Cycle.

Now break at least enough wires in N, according to some scheme for keeping the number of these breaks near a ninimum, so that there are no cycles left in the resulting net. Denote this result Nnc. Now Nnc has some component input and output leads dangling at the break points as a result of the breaks. Relabel these leads respectively as additional input and output leads of Nnc. Then Nnc is a well-defined Nc-net. Form the 2k+ 1, kc derivative of Nnc as specified in Theorem I. Then connect together those of Nnc's (2k +1) -wire, input and output variable bundles that correspond to leads which were originally broken in N. Then the result's bundle interconnection is identical to the interconnection of single leads in the original N. Denote this result N1.

Now in each of the loops in N1 which correspond to a cycle in N, there are one or more bundle-state resolver subnets. Each of these subnets has a delay of 2. Thus, in order to get N1 to operate in the same over-all manner as N, it is in general necessary to insert some delays in some of the loops of N1. This is always possible. For if nothing better can be done, d-1 delays can always be used at all component inputs which do not come from resolver subnets. This will compensate for the delays in all the resolver subnets.

Denote by N2 the most economical alteration of N1 which has all its loop delays properly fixed with respect to each other. Now if in realizing N2, the break points were chosen so as to minimize the cost of N2 (the number of bundle-state resolvers in N1 is highly important here), <u>let the corresponding form of N2 be denoted $CN^{\#}$.</u>

The Section is summarized in

<u>THEOREM IV:</u> Given any irredundant computer net, N, $CN^{\#}$ is a form of failure-correcting derivative 1) which has the same general organization as N, 2) which can correct any kf's that might ever occur in any of its enclosed loops and 3) which does not operate with any form of time-domain redundancy. $CN^{\#}$ also has the property that it can correct the effects of any set of k or fewer permanent component failures which might occur within it.

<u>CONJECTURE III</u>**: Given any irredundant computer net, N, $CN^{\#}$ is the most economical form of failure-correcting derivative of N 1) which can correct any kf's that might ever occur in any of its unclosed loops

** Assuming Conjecture II to be true.

(or which can correct the effects of any set of k or fewer permanent component failures which can occur within it), and 2) which does not operate with any form of time-domain redundancy.

REMARK: This conjecture is essentially based upon the belief that there do not exist any more economical over-all computer organizations than $CN^{\#}$'s. There may in fact be a completely unknown concept for computing with nets in which kf's are permitted to occur. And this concept may be more economical to realize than the one employed in $CN^{\#}$. However, granted the nonexistence of such a concept, $\overset{**}{\text{Theorems}}$ I and II, and factors similar to those mentioned just before the statement of $\overset{**}{\text{Theorem}}$ III, tend to lead one to the above Conjecture.

4.5 THE SECOND FORM OF FAILURE-CORRECTING COMPUTER NET

The remainder of this chapter describes the second form of over-all failure-correcting computer net that was mentioned in Section 4.1.

There are two variations of this form, just as there were the $CN^{\#}$ and $CN^{\#}$ variations of the over-all 2k+1 form. Again, the form is constructed in the form of N*. It is developed below in such a way as to show that one of its variations probably operates with maximum time-domain redundancy.

Both of the variations have the following constructional features in common: They are of the form of N*. Their M nets are entirely composed of the M_r type memory elements given in Theorem II. Their NC nets contain k+1 disjoint copies of their own corresponding irredundant realizations. The NC nets' individual outputs to their respective M nets are all sent over (k+1)- wire binary channels. The ith wire in each channel from NC to M, $0<i\leq k+1$, comes from the corresponding binary channel of the ith irredundant realization in NC. The general scheme is shown in Fig. 18, excepting that NC's input state resolver nets are most economical with delay d instead of as shown in the Figure. In Fig. 18, the F nets are the irredundant realizations corresponding to the over-all NC net shown. The y_i, $i=1, \ldots, n$, outputs of NC are fed into M, and the y_i, $i=n+1, \ldots, n+s$, outputs of NC produce the 0_j outputs of the over-all net. Since the inputs are presented to M on (k+1)-wire bundles, all the M_r elements of M have w_i's and u_i's equal to k+1. Thus, only homogeneous 1's can represent a bundle state of 1 on a bundle leading from NC to M. The outputs of M into NC are given over (2k+1)-wire binary channels. The states represented over these channels are resolved at NC's input, and then re-represented over (k+1)-wire binary channels. From there, the resolved inputs are combined with the I_j inputs, and the result is fed into the k+1 F nets as shown.

To continue, the NC nets of both variations of the second form of failure-correcting computer net operate into their respective M nets on the same sequential principle as was discussed for CN*. The timing problem that arises out of interconnecting the M and NC nets in Fig. 18 is the same as before with CN*. Hence it will not be further discussed here. Rather, the form in Fig. 18 will be considered to operate in the same manner that the operational model of CN* operated.

It is conjectured, in the light of Theorem**II, that a more economical N* form than the one shown in Fig. 18 could not possibly 1) at least detect kf's in its NC part, and 2) correct any kf's that might ever occur in any unclosed loop in its M part. The substantiation of this conjecture rests on the same principles that are contained in the proofs of Lemmas II and III and Theorem I. The proof of it however is blocked by the kd analog to the situation that blocked the proofs of Conjectures I and II.

** Assuming the truth of Conjecture II.

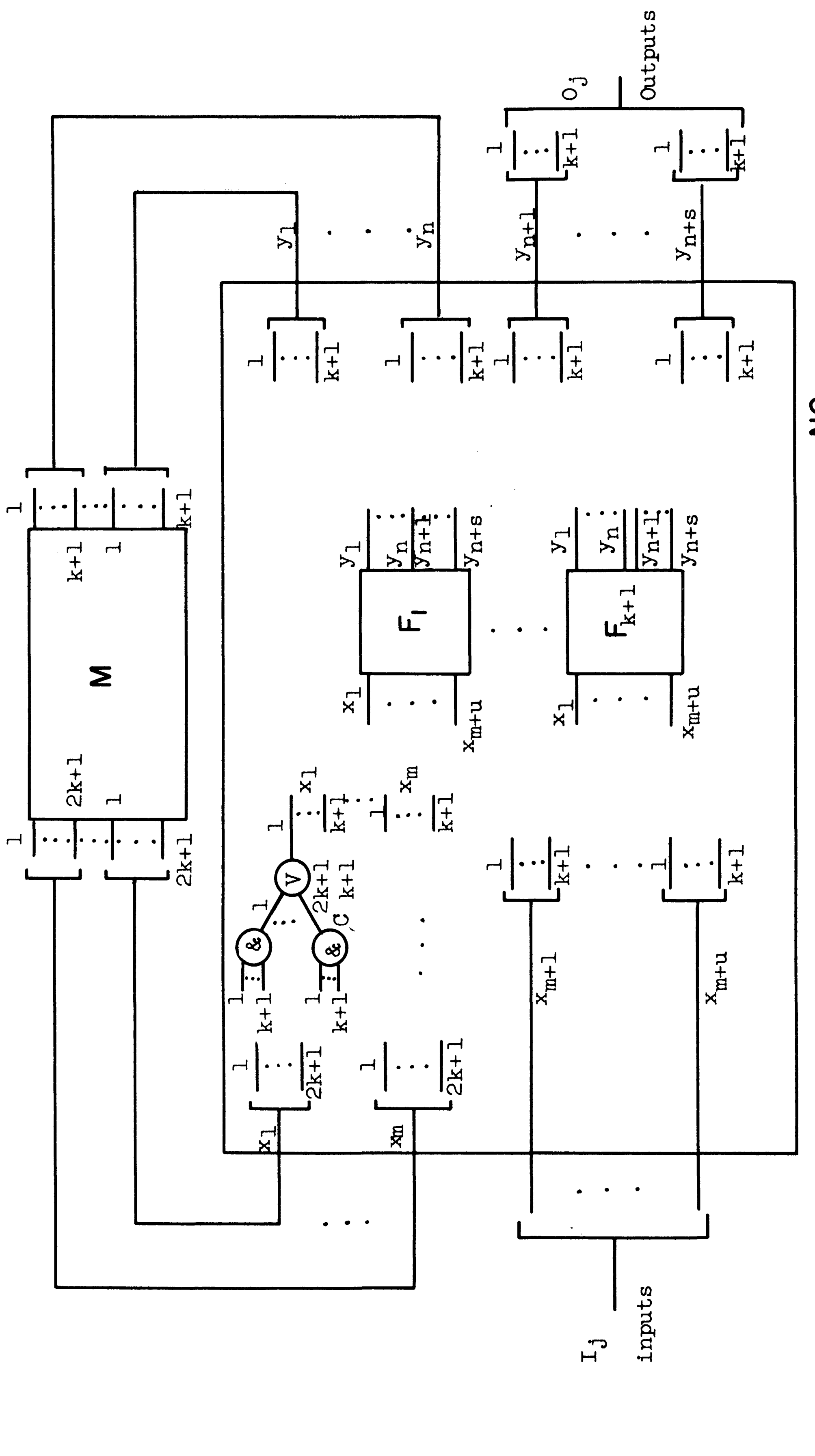

Figure 18. The Second Form of Failure-Correcting Computer Net.

The next thing is to determine if the form in Fig. 18 is sufficiently general. Suppose at some time t, z of NC's individual outputs into M ought to be 1's, but only u of them actually are, for some $u < z$, because of a kf in NC. Note here that since unanimous 1's have to be present on a bundle out of NC to produce any effect in M, z-u may be as great as k. Assume for the remainder of this Chapter that only jf's, $j \leq k$, are allowed in nc-nets and unclosed loops of over-all computer nets. Then u of M's stored quantities are respecified at t+d by the u correct inputs to M, but the other z-u quantities remain as before. Now if the form in Fig. 18 is permitted to continue operating in this manner, permanent mistakes will in general occur. So, some set of arrangements must be included to eliminate the possibility, B, of incorrect information from NC's being permanently registered in M. Two such sets of arrangements will be discussed below. The net containing the first set will be denoted DN*, and that containing the second, DN#.

The form of DN* is essentially the same as that shown in Fig. 18. The only difference is that DN*'s NC outputs are encoded in a k-error-correcting code so that M will receive the correct information regardless of whether kf's occur in NC and M or not. The details of the scheme are as follows: Suppose at some time t, the state of M is altered from S_{t-1} to S_t. Then NC interprets S_{t-1} at t-1 to be memory information state A_{t-1}, and S_t at t to be memory information state A_t. A memory information state is the information interpretation of the memory state. The explanation of the relationship between the S_i and A_i reads thus: The A_t represent the proper sequence of information states in the M block of DN*, (with respect to some initial state A_0 of M). The S_i are altered in sequence by NC so that NC interprets them as the successive A_i. Thus all the Si states which occur when DN* operates without any failures must have memory-state-point neighborhoods about them consisting of at least all the state points distant[1] less than or equal to k from them. And all the points in each such neighborhood must be interpreted by NC as the corresponding A_i. For only bundle-state 1's out of NC can cause changes in M, and for every output of NC there exist kf's such that k of the bundle states which ought to be 1's are not. Consequently, NC must be able to alter any memory state point in A_{t-1}'s memory-state-point neighborhood to at least some point in A_t's memory-state-point neighborhood. This of course is possible to arrange from the familiar encoding and decoding concepts of classical coding theory. For the situation here is identical to the one there if the M net and connections are thought of as a noisy channel, and NC is regarded as a perfect decoder-encoder unit (that is, its failure structure is transferred to the channel).

DN* suggests a whole class of forms which generalize on its scheme of operation. Each member is the same as DN* except that the w_i's and u_i's of its M_r's range in general from k+1 to 2k. Evidently, the operation pattern of these forms is described by the above discussion if the neighborhood radii[2] there are decreased from k/1 to $-\lceil -k/2 \rceil, \cdots, -\lceil -k/k \rceil$ as k+1 there is increased from k+1 to k+2, $\cdots$, k+k = 2k. For with (k+i)-wire bundles leading

1. See the end of Chapter 2, a footnote defining distance.

2. A neighborhood radius is 1/2 the maximum distance between any two points in a state point neighborhood.

from NC to M, each bundle state is correct at least for all (i-1) f's in NC; and (i-1) equals 1, 2, . . . , k as k+1 is increased from k+1 to 2k. Denote the generalized form of net representing the above class of forms, D*N*.

Note that the NC part of D*N* is effectively kc, and that its failure-correcting redundancy is still in the equipment domain. The scheme of operation of D*N* is conjectured to represent the most economical meet**, within the general form of D*N*, with the requirement for the elimination of B. This conjecture is made provided the elimination must be accomplished with a kc, NC net.

At this point, from Theorem**III there follows
CONJECTURE** IV: Theorem**III holds with "CN*" replaced by "Either CN* or D*N*", and with 4) deleted.

Before moving on into the discussion of the DN# form, note the general extra costs incurred in altering DN* to D*N* to eliminate B. They include the costs of extra M_r elements, and A_i resolving and output encoding nets in NC. On the other hand, D*N* is cheaper than the over-all 2k+1 form, CN*, 1) at the inputs to M, and 2) to a certain extent, because it only has (k+1)-fold duplication in its NC part. 1) is because the w_i's and u_i's of the M_r's of D*N* are all less than 2k+1.

Evidently, the neighborhood idea used in D*N* is one version of the idea that was dismissed at the end of Chapter 2 for isolated kc nets. It is not dropped here because NC's output must feed into a memory net, which adds a condition that was not present in Chapter 2. Still another version of the idea at the end of Chapter 2 which has been dismissed here, however, entails single resolutions of M's complete memory states instead of separate resolutions of each of M's M_r states.

** Assuming Conjecture II to be true.

4.6 THE DN# VARIATION OF THE SECOND FORM OF COMPUTER NET, AS ARRANGED FOR TRANSIENT TYPE FAILURES

This Section discusses the second set of arrangements that can be added to the general form in Fig. 18 to eliminate B. The first set, incorporated above into D*N*, and this set, employed below in $DN^{\#}$ and $D^{\#}N^{\#}$, seem to exhaust all the really promising possibilities.

First of all, $DN^{\#}$ is developed in the present Section to cover the case where only transient type failures are allowed. The essential feature of this case with regard to the satisfactory operation of $DN^{\#}$ is the assurance it gives that $DN^{\#}$'s NC net will not produce failure-corrupted outputs infinitely many time instants in succession. In the next Section, the general case where either permanent or transient type failures are allowed is considered; and with this consideration $DN^{\#}$ is increased to another form, $D^{\#}N^{\#}$, to compensate for the change in failure structure.

Allowing only transient type failures for the remainder of this Section then, consider the following restriction on the operation of the form in Fig. 18. Suppose the NC part of that form were only permitted to alter the M part's state by 1 bit at a time; and that each successive alteration of M's state had to be successfully completed in order for M to specify its next state. With this requirement, if a jf, $j < k$, occurs to corrupt an NC output, all of NC's corresponding output bundle states are interpreted by M as 0's. Hence there is no change in M's state, and NC's next attempt at altering the state of M merely constitutes a repetition of the previous attempt. In this manner, successive re-attempts continue until one of them is successful, and then NC begins its attempt to cause the next regular state change in M.

Such a pattern of operation is easy to instrument in the form of Fig. 18. Unfortunately, however, the result is extremely uneconomical if its operation is arranged so as to make it sufficient to represent a general purpose computer (or, abstractly speaking, a completely finite Turing machine[1]).

The reason is as follows: a general purpose computer must be able to carry out sets of instructions (perform mathematical functions) on arbitrary sets of data (variables). Now, suppose A is to be added to B in the above form where A and B are binary representations of natural number quantities. Then the first sum bit that NC specifies to M may be either a 1 or a 0, depending upon the values of A and B. In one of these cases, the sum bit specification does not change the state of M, since the memory location where the bit is sent must be in either the 1 or 0 state before its arrival. But if there is no change in the state of M, the over-all net reaches an equilibrium state and therefore cannot further compute. The only way out of this difficulty, which does not require the single-bit computing concept to be discarded or the basic form of the net to be altered, is to let all of NC's outputs change the memory state in the same direction: that is, always change 0's to 1's or vice versa. Then the locations in M where NC's output 1's are stored must be used to register successive results during computations. But if this scheme is used, the number of 1's stored in M monotonically increases with time. This requires that a huge number of memory locations be available in M. Otherwise the over-all is not able to perform general types of programs—especially those involving slowly converging iterative

1. See the end of Section 4.2

routines. Thus it is apparent without further investigation that the above form would cost at least more than either D*N* or CN* for the same amount of general computing ability. Since the point of this Section is to derive a form which at least might be more economical than D*N* and CN*, the above form will not be discussed further.

If the single-bit computing concept is to be salvaged with any prospects of economy, the form in Fig. 18 must be altered at least as follows: There must be an arrangement for NC to put out successive single bits to M such that these successions amount to the production of q-bit outputs from NC to M, $q > 1$. This requires M to be partitioned into disjoint nets, M1, . . . , Mq and successive of the single bits to be sent into the different Mi in sequence. In order to achieve this, after (or with), each successful 1-bit transmission from NC to Mi, the state of some one of M's memory locations must be reliably changed in order to indicate that the next bit from NC to M must go to M(i + 1), or M1 if i=q. Special locations in M must be reserved for these changes, for it was seen earlier that computation bits from NC to M may not always produce changes in M's state.

Consider the case where q=2. The schematic of a net which is sufficient for this case is shown in Fig. 19. The notation in Fig. 19 is consistent with that in Fig. 18, and in anticipation of its adequacy the over-all net form in Fig. 19 is denoted DN#. M_o in the Figure is an M_r element with r= 0. It is in M_o that the indicator location mentioned above is contained. M_o's set inputs consists of all the set and reset inputs of all the memory elements in M1; and its reset inputs consist of all the set and reset inputs of all the memory elements in M2.

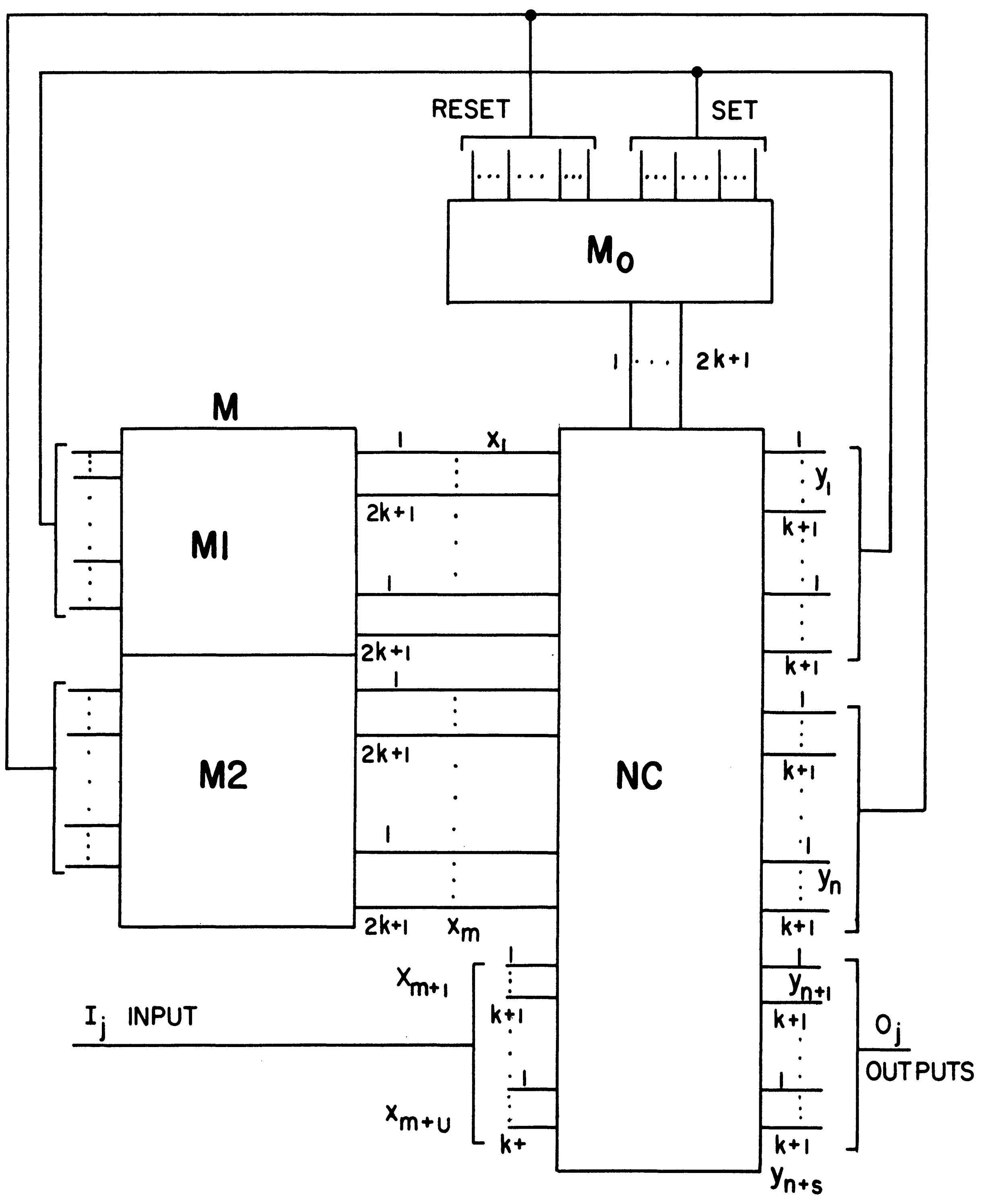

Figure 19. A Schematic of $DN^{\#}$.

The relative timing among the memory locations in Fig. 19 is arranged as follows: When the state of M_o directs NC's operation into M1, NC s subsequent 1-bit output changes M_o's state so that the resulting M_o state directs NC into M2. Then NC's next output resets M_o to its previous state, causing NC to work into M1 again. This pattern of operation is continued until the net finishes the computation it is performing. At any point, if a jf, $j \leqslant k$, occurs to cause one of NC's outputs to be spoiled, successive re-attempts are made to produce that output until one is successful before M_o's state is changed.

Since the outputs of M are continuously fed into NC, the following timing problem might arise in $DN^{\#}$: The delays from NC's input to several of its outputs might vary so much that successive bits from one output might persist long after bits from the next couple of outputs had been produced. To illustrate this with an example, suppose NC is a gnc-net, and that the delay from NC's input to y_1 is 100 time units, while that to $y_2, \cdots, y_n$ is only 5 time units. Then after the first successful bit out of y_1 to M1 occurs, it may take up to 99 more time instants to clear NC of the effects of those inputs from M which produced the y_1 output. For M's outputs are continuously fed into NC. Now NC's next output, say from y_n to M2, may be produced without any failure. If so, as soon as y_2 stops putting out 1's, the continued outputs from y_1 set M_o to 1 again whether NC's next output to M1 is correctly produced or not. Thus a permanent mistake might occur at this point.

From the example, it is apparent that a very general way of solving the problem here is to make the delay from NC's inputs to all of its outputs the same. Hence the general form in Fig. 19 remains as what is desired for $q = 2$. Note that

M cannot be vacuous in this form as it can in $CN^{\#}$ and D^*N^*.

The only additional point here is that, somehow, it must be known when I_j inputs to $DN^{\#}$ have been correctly stored. There are many ways of reliably doing this. The simplest way is to bitwise store all binary word inputs to $DN^{\#}$ at one set of locations in M as follows: Run all the set and reset inputs to those locations into a $2k+1$ type "or". Then as each bit from $DN^{\#}$'s input is correctly stored, the majority output of the "or" net switches from 0 to 1 for one time instant. This switch constitutes a reliable signal to the outside world that the last bit was properly stored, and that the next bit can be presented for storage. The detailed sequence for carrying out a "store" order in $DN^{\#}$ is not taken up until later.

Now a heuristic will be given to show that the form in Fig. 19 is sufficient to represent any completely finite Turing machine. It will be shown that $DN^{\#}$ can perform any arbitrary sequence of combinational functions (instructions) on arbitrary sets of arithmetic variables (data). It will be assumed that all functional and variable values are represented by binary sequences.

Let the ordered sequence of instructions $DN^{\#}$ is to carry out be successively stored in memory locations $1, \cdots, z$ in M1. Also reserve $-[-\log_2 z]$ memory locations in M1 to store a count configuration between 0 and $z-1$ inclusive. Denote these latter locations the <u>instruction counter, or IC</u>. Let the state of IC specify at each point which of the z instructions NC is to work on. Arrange to have all the successive count representations in IC from 0 to $z-1$ differ by only 1 binary digit[1].

Let the data and instruction word lengths in $DN^{\#}$ be denoted by w. Let M2 contain $-[-\log_2(w+1)]$ memory locations to store a count configuration

1. This counting scheme is the classical <u>"Grey Code"</u> scheme.

between 0 and w inclusive. Denote these locations <u>the digit counter, or DC</u>. Arrange NC so that at any point in a computation, if the state of DC represents a number j, where $0 < j \leq w$, the jth binary digit of the function value NC is computing is the next one to be sent to Ml. Let all the successive count representations in DC from 0 to w differ by only 1 binary digit.

Then, after the starting mode of $DN^{\#}$ has been completed (the reading in of its data, etc.) the first instruction stored in Ml can be carried out as follows: First the states of IC and DC are 1, and the state of M_o is 0. Then NC observes the first instruction, which is a binary representation of the form: f, Ll, $\cdots$, Ls, L(s+1). This representation means "perform operation f on the variable values at the sets of memory locations Ll, $\cdots$, Ls, and store the result at the set of memory locations L(s+1)". Here the Li may be any of the sets of variable value memory locations in Ml. As soon as NC is able to correctly transmit the lst bit of f's value to Ll, M_o's state is changed to indicate this. NC then changes the state of DC to represent 2. Next, NC sends the second digit of f's value to Ll. After this it counts up one more in DC. Then NC sends f's third digit to Ll. This pattern of operation is continued until the w th digit of f's value is sent to Ll. Then the count state of DC is returned to 0. (If $-[-\log_2 (w+1)] \neq \log_2 (w+1)$, this must be done in successive steps. These steps can be arranged by having NC produce l's to only the set-input side of M_o when the count state of DC is greater than w.) After this the count state of IC is increased to 2. Then by increasing the count state of DC to 1, the operation cycle begun above is complete. So the operation sequence of $DN^{\#}$ repeats until all z instructions in Ml have been carried out, and the results of the computation have been read out on the O_j output wires.

Now another word about the starting mode in $DN^{\#}$ is in order. This mode can be taken care of by fixing an initial set of wire states for $DN^{\#}$ to start from, and wiring in enough store instructions to enable it to resume operation from a programmed set of instructions.

The above completes the sufficiency demonstration for $DN^{\#}$. It seems probable that all the features that have been given $DN^{\#}$ are necessary for an economical and sufficiently general N* net not having a kc NC part (as in CN* and D*N*).[1] In fact $DN^{\#}$ is the result of an attempt to realize a failure-correcting N* form which has a maximum amount of redundancy in the time domain. Now the fact that $DN^{\#}$'s memory is only partitioned into 2 parts does not constitute an economic restriction. For if $q > 2$, each set and reset input of every element in Ml, $\cdots$, Mq must still go to at least one NC —output— indicator memory location, such as in M_o of $DN^{\#}$. But these extra memory locations can just as well be put in Ml, and all the Mi's but one lumped into one memory and redesignated Ml. Hence, including M_o in M because of the closed loop it contains, we have

<u>CONJECTURE** V:</u> Theorem** III holds with "CN*" replaced by "Either CN*, D*N* or $DN^{\#}$"; 2) replaced by the condition "which allows kf's to occur in its NC part without a permanent mistake's ever occurring"; 4) deleted; and 5) replaced by the condition "in which only transient type failures ever occur."

At this point it is felt that some final remarks should be made concerning 1) the cost of $DN^{\#}$, and 2) the list of instructions that its NC block might be

** and 1. Assuming Conjecture II to be true.

designed to perform. With regard to 2) the following list of instructions is both sufficient and very economical to realize in NC[1]: 1) S, L1, L2: store the contents of memory location L1 at memory location L2. Here, L1 can be the input and L2 the output of $DN^{\#}$; 2) A, L1, L2: add the contents of L1 to those of L2 and store the results in L3; 3) C, L1, L2: take the 2's complement of the contents of L1 and store the results in L2. This instruction together with 2) enables NC to subtract; and 4) C, L1, N: change the state count of IC to N if the contents in L1 are 0. Otherwise do nothing. This instruction permits $DN^{\#}$ to perform iterative type programs without having each single instruction that it carries out stored in a different memory location.

Now consider the cost of $DN^{\#}$. It may be compared in a qualitative sense to that of $CN^{\#}$ and D*N*, since all of these forms can use essentially the same general schemes for computing. The extra cost of $DN^{\#}$ over that of the form in Fig. 18 comes from M_o, DC, and the extra subnet that is needed in NC to enable it to work with M_o and DC. $DN^{\#}$'s saving over the cost of CN* is the same as was mentioned for D*N*. On the other hand, $DN^{\#}$ has none of the extra costs that were mentioned for D*N* over the form in Fig. 18.

The following conjecture is known to be true for fairly small k and for NC nets of fairly few input variables. The proofs that are available for these special cases will not be given because of their lengths. However, the conjecture below seems just short of obvious from them.

---

1. The argument for this assertion is not given here because of its inherent length.

CONJECTURE VI: Conjecture **V is true with "Either CN*, D*N*, or $DN^{\#}$" replaced by "$DN^{\#}$".

4.7 THE $D^{\#}N^{\#}$ VARIATION OF THE SECOND FORM OF COMPUTER NET, AS ARRANGED FOR TRANSIENT AND PERMANENT TYPE FAILURES.

The purpose of this Section is to extend $DN^{\#}$ in some economical fashion 1) so that it can correct kf's in its NC part even if permanent type failures are allowed, and 2) so that the result is neither CN* nor D*N*. Here, 2) is in the interest of exhausting the set of forms which are possibly most economical.

The following is apparent if $DN^{\#}$ is to be extended to a form, $D^{\#}N^{\#}$, in accord with 1) and 2) above. $D^{\#}N^{\#}$ must have at least k disjoint standby copies of the subnet, call it S_i, between $D^{\#}N^{\#}$'s NC net inputs and the ith wire in each of its output bundles. Here i is not specified since all the S_i are identical anyway. These subnets must be switched into operation in the event of permanent failures in any of the S_i, and the corresponding S_i must be cut out. If the latter is not done, eventually (2k +1)-wire bundles will have to be fed into M. And then the whole saving of the (less than 2k+1)-wire bundle input to M arrangement will be gone. To continue then, in order to be safe, provisions must be made for $D^{\#}N^{\#}$'s correct operation regardless of which k of the set of k +1 S_i's and k standby S_i nets fail. Finally, if a standby net is mistakenly switched in for an S_i which contains only transient failures, the mistake must not be allowed to persist through all later time. Since transient failures can occur in the same component indefinitely many times in succession, this requires a periodic (with either failures or time) and complete switch-in, cut-out pattern for the S_i's.

** Assuming Conjecture II to be true.

When a failure occurs in $DN^{\#}$ to corrupt an NC output, there is no effect produced to indicate this. Consequently, if standby nets are to be switched in for S_i's which fail, either a reliable clock-controlled (i.e., time basis) failure indicator must be available, or else the switch-ins and cut-outs must occur in a regular periodic (with time) sequence regardless of whether failures occur or not. For any other means of obtaining a reliable failure indication must employ straight failure-correcting logic in NC, perhaps of the type found in CN* or D*N*. But this simply extends $DN^{\#}$ to one of these forms, and so is not what is wanted here.

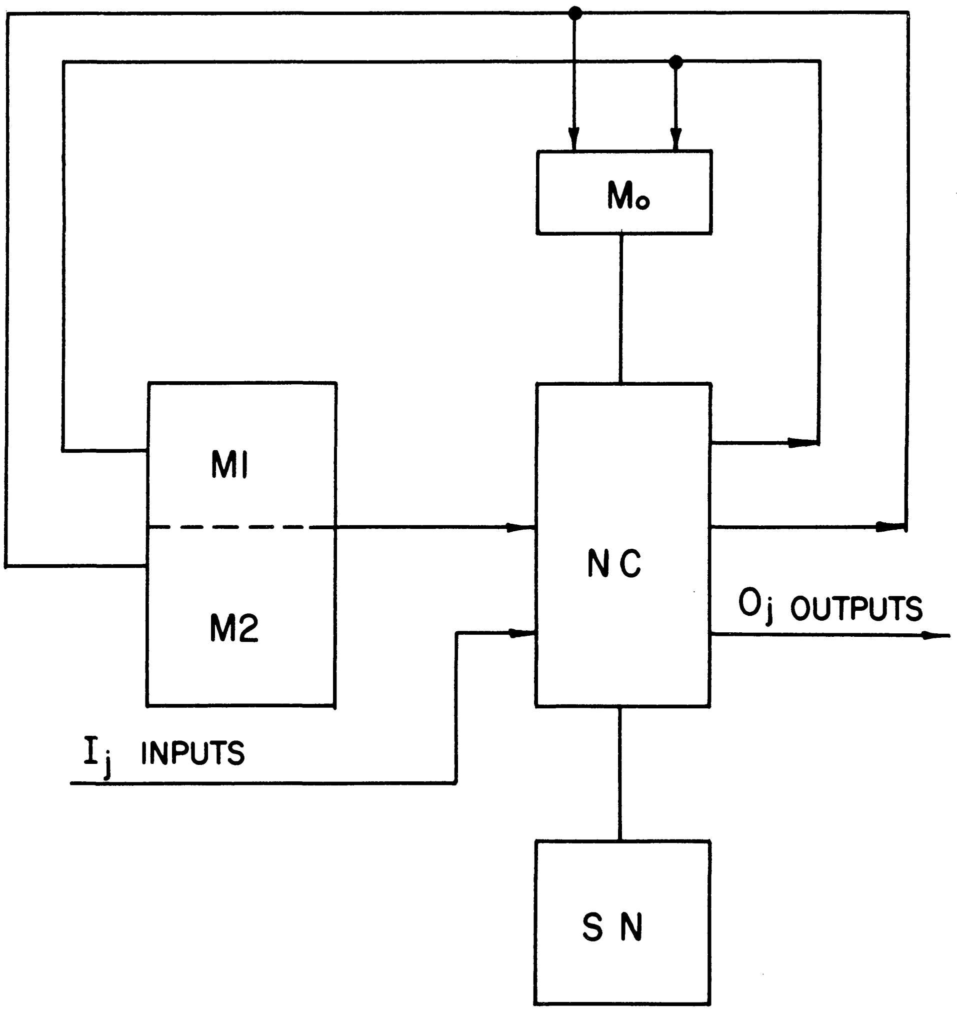

Figure 20. $D^{\#} N^{\#}$.

Fig. 20 shows the schematic of an over-all extension of $DN^{\#}$ which just meets the requirements specified above. The single lines between the blocks in Fig. 20 are merely to indicate corresponding communication channels. Also, with one exception, the notation in Fig. 20 is identical to that used for $DN^{\#}$ wherever the same symbols are used. The exception is NC. NC in Fig. 20 differs from NC in Fig. 19, only 1) by having k standby S_i's to switch into operation if some of the original S_i's fail, and 2) by having a subnet to admit NC's outputs to come from only the (k+1) subnet of S_i's and standby S_i's that SN's output specifies. The particular arrangements in NC which realize this admission remain unspecified, but it will be assumed that they are most economical. The specifications that are given for the SN block in Fig. 20 are as follows: SN has u different outputs that it can transmit to NC, and each time instant its output to NC is one of these. SN's outputs are periodic, and exhaust all u possibilities every period. Since SN's output to NC determines which set of standby S_i's are to be switched in for which set of original S_i's, u must be at least as great as all such switching possibilities. This number is $\sum_{i=1}^{k} (C_i^{k+1})(C_i^{k})$ which is easily derived by considering the switch-in sequence shown in the right hand column of Table 4. In the Table, the rows in the left hand column list in order the i's of the successive sets of S_i's to be cut out when the set of standby S_i's (also indicated by their i's) in the right hand column is switched in. To continue with SN's specifications, all of its outputs are synchronized with the outputs of M and M_o as follows: they reach NC's input at times which enable them to specify which of the 2k +1 S_i's and standby S_i's in NC are to produce the corresponding set of

TABLE 4. The cut-out and switch-in sequence of the S_i's in $D^{\#}N^{\#}$.

	Cut-out sets	Corresponding Switch-in sets	
each row contains all singles	1, 2, . . ., k+1	1	
	1, 2, . . ., k+1	2	
	.	.	all
	.	.	singles
	.	.	
	1, 2, . . ., k+1	k	
each row contains all pairs	(1, 2), (1, 3), · · ·, (k, k+1)	(1, 2)	
	.	.	all
	.	.	doubles
	.	.	
	(1, 2), (1, 3), · · ·, (2, 3, · · ·, k+1)	(k - 1, k)	
		.	
		.	
		.	
each row contains all k-tuples	(1, 2, · · ·, k), · · ·, (2, 3, · · · k+1)		

outputs from NC to M. The minimum duration between output changes in SN is also restricted by similar timing considerations. Finally, SN is a most economical net which operates reliably in accordance with the above specifications[1].

With the above provisions, $D^{\#}N^{\#}$ operates as required. For provided there are no more than k permanent failures in NC at least once every one of SN's periods, there will be a set of k+1 S_i's and standby S_i's that attempt to produce a 1-bit output from NC to M with a non-zero probability of success. Hence, as the number of SN's periods grows large, the probability that NC will produce a 1-bit output to M approaches 1.

The extra cost of $D^{\#}N^{\#}$ over that of $DN^{\#}$ comes from SN and the extra subnets in NC. Apparently, from above, when permanent type failures are allowed these costs are all necessary. The big saving in the cost of $D^{\#}N^{\#}$ over that of CN* and D*N* follow from the cost discussions at the end of Sections 4.5 and 4.6.

<u>CONJECTURE** VII:</u> Theorem** III holds with "CN*" replaced by "Either CN*, D*N*, or $D^{\#}N^{\#}$"; 2) replaced by the condition "which allows kf's to occur in its NC part without a permanent mistake's ever occurring"; and 4) and 5) deleted.

As a last remark with regard to $D^{\#}N^{\#}$, $D^{\#}N^{\#}$ may easily be extended so that the output of SN changes only if a failure occurs to corrupt an NC output. With such an arrangement, and if the switching sequence in NC is as given in Table 4, SN's output can be used as an indication of which of the original S_i nets have failed. The arrangement can be achieved quite easily roughly as follows:

1. With respect to the use of the word "reliably" here, recall the underlined assumption given after the explanation of Fig. 18.

1) let SN be a CN* or D*N* type cyclic, Grey-code, bidirectional counter; 2) add a clock, C, of the form shown in Fig. 16 which a) puts out a 1 to SN to reliably increase its count state every t instants (t greater than all closed loop delays in the over-all net) and b) starts a logical performance in NC at these same instants; 3) effectively pass all the inputs to M_o through a 2k+1 "or" by tapping off M_o's two "or" groups and let the output be fed into SN to reliably decrease its count state by 1 if NC puts out a 1-bit output to M. Then when trials are unsuccessful in NC, SN's count state is permanently increased by 1, otherwise not. Since SN is cyclic, when its count state gets to u, its next state will be 0. This insures that no switch-ins for transient failures will ever persist throughout all later time.

CHAPTER 5

kd NETS

5.1 INTRODUCTION

Up to this point, only failure-correcting types of nets have been considered. Hence it is of interest to have something formulated with regard to kd nets[1].

The purpose of this Chapter is to consider ways of obtaining economical kd derivatives of irredundant nc-nets. A kd derivative of an irredundant nc-net, N, is a kd net which realizes the same function as N. The problem of this Chapter stated more precisely then is: given any nc-net, N, which realizes the set of functions, F, from the input scheme, S, what is N's most economical kd derivative which also realizes f from S. Here, "input scheme" refers to the manner of presenting variable-value inputs to N. For example, N's input variables might be represented over different numbers of wires and/or by binary sequences, etc.

With the net formation allowances that were made in Chapter 1, the problem as stated is presently rather closed. For, apparently, when redundancy insertion is restricted to the equipment domain, k+1 disjoint realizations of N constitute a very economical kd derivative of N. In fact, this derivative seems to be most economical in the sense indicated, but as yet the author has been unable to prove it for cases where more than k+1 output wires are permitted. When only time-domain redundancy is allowed, binary sequences of the type discussed in Section 2.4 are used in N. The spectrum of possibilities that lie in between the

1. See the definition of a "kd net" in Section 1.3.

above two redundancy-type extremes follow immediately by proportionately mixing the forms of redundancy found in them.

While the above schemes are acceptable, they are not really satisfactory in many respects. For example, for some applications it might be that the O_j output channels of the forms in Chapter 4 were only allowed to be transmitted over m-wire channels, $m < k+1$. In such cases it would be necessary to know what additional net-formation admissibilities were needed to enable kd outputs to be produced over $(m < k+1)$-wire channels. This question is considered at the beginning of the next Section.

5.2 DOMINANT AND RECESSIVE LOGIC

There remains only one uninvestigated class of net formations for cases where only unilateral components are allowed. The distinguishing characteristic of this class is defined by the following adjunction to the component interconnectability rules given in Chapter 1: Let it be permissible to merge the output wires of n different components into a single wire, for any finite $n > 0$. Let this wire be in the 0 or 1 state according as all of the merged wires are in the 0 state or at least one of them is in the 1 state respectively. Functionally, a merger point behaves as a perfectly reliable "or" component, except that there is no delay associated with its operation. Denote any net which contains such a merger a dominant and recessive logical net, or DRL net[1].

The merging rule just defined is not of only academic interest: For a physical correspondent to an abstract merger point can be realized as a solder joint in an electronic digital computer. In this case the 1 state corresponds to

1. The dominance of the 1 state and the recessiveness of the 0 state are responsible for the terminology.

ground and the 0 state to non-zero potential. From such a point of view, the perfect reliability ascribed to merger points seems justified. For if computer-circuit solder joints are well-formed, they almost never fail unless they are subjected to very severe environmental conditions. And in these cases, special wiring techniques can often be found to insure that they are sufficiently reliable to justify the assumption.

On the other hand, physical realizations of abstract mergers might cause some difficult loading problems. For example, the circuits in the basic logic packages of most electronic computers would have to be redesigned if mergers were permitted. Thus, in order to cut down on the number of mergers, the following arbitrary restriction will govern their use from here on: mergers will only be permitted at the outputs of nets which have fewer than $2k+1$ wires in their binary output channels. It can easily be shown that this does not constitute any restriction on the set of realizable output characteristics of allowable nets.

5.3 THE USE OF THE MERGER TO PRODUCE kd NETS

At this point, the problem of specifying kd nets from their corresponding irredundant realizations is reconsidered. Until it is stated otherwise, only equipment-domain redundancy will be allowed for failure-detecting means. The use of the merger allowance with respect to the kd net problem is best illustrated with an example. Fig. 21 shows an irredundant realization of the function $f(x, y) = x' \cdot y \vee x \cdot y'$. Let the example problem be to find a 3d net which also realizes $f(x, y)$ and has only 2 wires in its binary output channel.

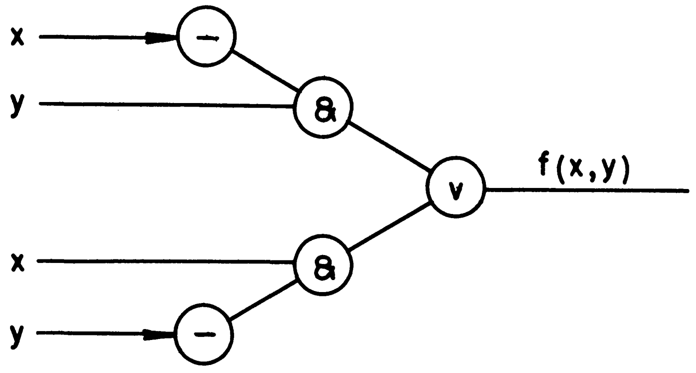

Figure 21. A nc-net.

Now consider the net and net input scheme shown in Fig. 22. Throughout the remainder of the Chapter assume that all net input schemes are similar to the one shown there. That is, in every case, assume that all the net's input variables and their negations are perfectly reliably presented to the net over separate wires. The justification for this assumption will be given later on.

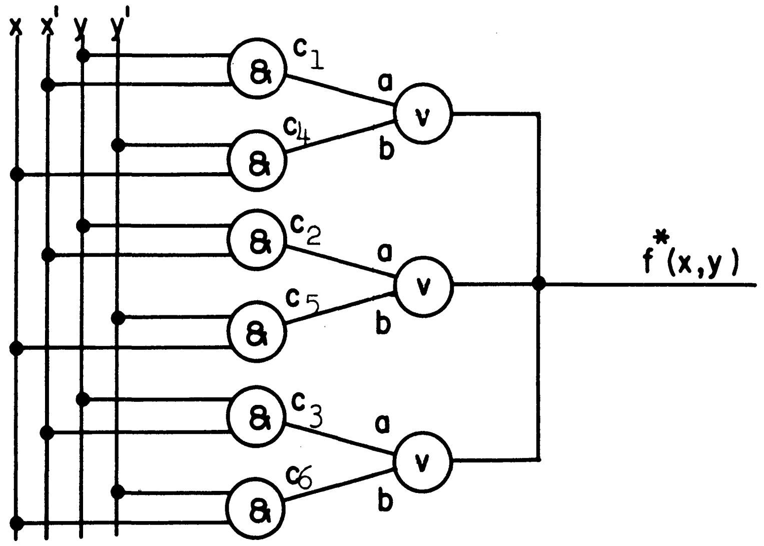

Figure 22. A Merged Output Net.

The states of the net in Fig. 22 at points "a" are 1 at time t-1 if x is 0 and y is 1 at t-2, and if no more than two failures occur among the components c_1, c_2, and c_3 at t-1. Also, the states of the net at points "b" are 1 at t-1 if x is 1 and y is 0 at t-2, and if no more than two failures occur among the components c_4, c_5, and c_6 at t-1.

Similarly, if either x is 1 and y is 0, or x is 0 and y is 1 at t-2, and no more than any combination of two component failures occur in the net from t-1 to t, the output of the net, labeled $f^*(x,y)$, is 1 at t. Thus the net's output must be 1 at least at all those times when it ought to be provided no more than a 2f occurs.

However, the output state of the net in Fig. 22 can still be 1 at some time t when it ought not to be if a single component failure occurs at the first level at t-1 or the second level at t. Now by the formulation of the problem, only one more output wire can be added to the present net to make it 3d. This can be accomplished by adding a merged output realization of $f'(x,y)$ as shown in Fig. 23. The net there has the following characteristics: If both its output wires are 1, nothing is known about the corresponding input. However, it is known that at least a 1f occurred in the half of the net which ought to have produced the 0 output.

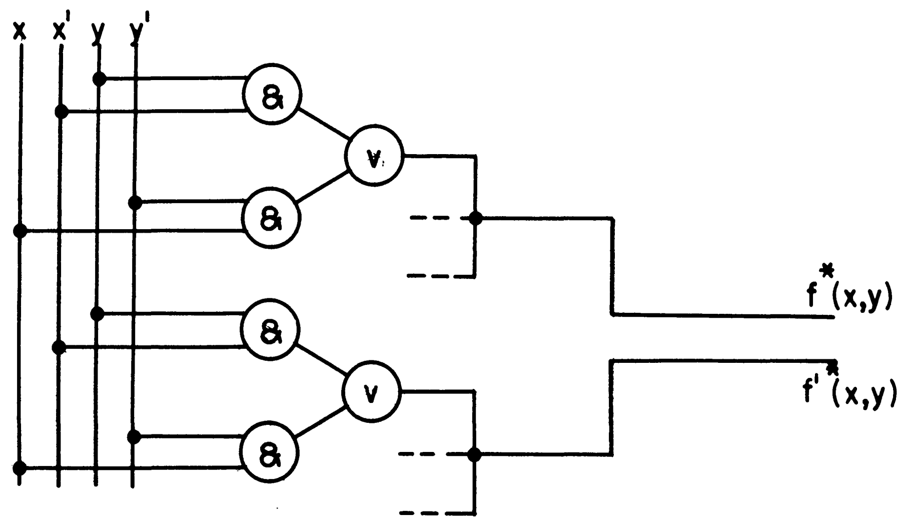

Figure 23. A 3d, 2-Output-Wire Realization of $f'(x,y)$.

If both of the net's output wires are 0, again nothing is known about the corresponding input, but it is known that at least a 3f occurred in the half of the net which ought to have produced the 1 output. Finally, if one of the net's outputs wires is 0 and the other 1, the state of the f^* wire must represent the correct state of

$f(x, y)$ at least if no more than a 3f occurred in the over-all net during the determination of the output. Hence the net in Fig. 23 is a 3d derivative of its corresponding irredundant form. Note also that it is always 2c in the side which produces the 1 output.

Now that the general idea of using mergers to construct kd nets has been introduced, it will be generalized. First of all, define a kf* net to be one which has only 1 output wire, where that wire comes from a merger of k irredundant realizations of $f(x_1, \ldots, x_n)$. (It is assumed in this definition that the input scheme is always the same as the one defined above for the present Chapter). Then we have

LEMMA V: The minimal costs of equi-delay f* and f'* nets are equal. This is most easily seen as follows: Let there be given a realization of $f(x_1, \ldots, x_n)$ which has minimal cost for some delay d. Then this realization can be represented uniquely, without thought to time delays, by an expression in the variables $x_1, \ldots x_n$ and the functionals "v", ".", and "'". Also, sets of parentheses must be used in the expression to point off its distribution. Now suppose terms of the form $(x_1 v \ldots v x_i)'$ and/or $(x_1 . \ldots . x_j)'$ are contained in this expression. Then these terms can just as well be written in the form $(x'_1 . \ldots . x'_i)$ and $(x'_1 v \ldots v x'_j)$ respectively. But by the assumed availability of x_i and x'_i inputs to the net, for all i, the net defined by substituting in the expression the forms $(x'_1 . \ldots . x'_i)$ for the forms $(x_1 v \ldots v x_i)'$ and the forms $(x'_1 v \ldots v x'_j)$ for the forms $(x_1 . \ldots . x_j)'$ is a more economical delay-d realization of f than the given one. Also its timing characteristics are the same as the given one's because "not" components have zero delay. But this contradicts the hypothesis, so no terms of the form $(x_1 v \ldots v x_j)'$

or $(x_1 \cdot \ldots \cdot x_j)'$ can be contained in f.

By an easy extension of this argument, it can be concluded that in general no negated parentheses can be contained in the original expression. Consequently, no "not" components can be contained in an irredundant realization of f or in the corresponding f^* net. From this result, and the argument which supports it, it is apparent that a delay-d f' net is defined from the representative expression for a minimal cost f net by substituting primes of variables for variables and vice versa, and "and" functions for "or" ones and vice versa. Thus by symmetry, the minimal costs of f and f' nets of delay d are the same. Hence also the minimal costs of equi-delay f^* and f'^* nets are the same.

Before the main results of this Chapter can be stated, some more definitions must be given. Define a $kf^*f'^*$net to be one which has only 2 output wires, where one comes from a kf^* net and the other from a kf'^* net. Let $\mathcal{F}$ represent the set of binary functions on binary variables $f_1(x_{1_1}, \ldots, x_{n_1}), \ldots, f_s(x_{1_s}, \ldots, x_{n_s})$.

Then let a $k\mathcal{F}^*\mathcal{F}'^*$ net be one which has s pairs of output wires where the ith pair comes from a $kf_i^*\,f'^*_i$ net, $i=1, \cdots, s$, and such that the over-all net is a most economical one which has this property

5.4 SOME RESULTS ON MOST ECONOMICAL kd NETS.

The purpose of this Section is to develop some results pertaining to most economical, m-wire-output, kd nets, $m=2, \ldots, 2k$. The first result covers the case for m=2, and the second and third results cover the cases for $m > 2$. All the discussions of this Section assume the following: the merger is admissible, the input scheme is as underlined in the previous Section, and only equipment-domain redundancy is allowed.

<u>THEOREM V</u>: Let there be given any irredundant nc-net, N, which realizes the s functions $\mathcal{F}$. Then the corresponding $k\mathcal{F}^*\mathcal{F}''^*$ net, $k>0$, is the most economical kd derivative of N from which each individual output is produced separately over not more than 2 wires. If the cost of N is c, the cost of $k\mathcal{F}^*\mathcal{F}''^*$ is 2kc.

<u>REMARK</u>: Note that by past definitions, the following is immediate: each individual output must come from a kd realization of one of the functions $f_i(x_{1_i},\dots,x_{n_i})$, $i=1,\dots,s$, and the binary output channel over which it comes must transmit at the rate of 1 bit per time instant.

<u>PROOF</u>: First of all, it is known from the last Section in Chapter 1 that kd nets cannot produce individual outputs over single wires. Hence 2 wires are necessary.

Now each wire in any 2-wire binary output channel can go to an incorrect state in at least one direction (i.e. from 1 to 0 or from 0 to 1) for some 1f. This is true whether the wires come from mergers or not. Consequently, only 10 and 01 can be used as perfect 1, 0 configurations over 2-wire binary output channels which come from kd nets. For to illustrate what might happen if these perfect 1, 0 configurations did not differ in both places, consider the following example: Suppose the configurations 01 and 11 were used as the correct representations of 0 and 1 respectively over a 2-wire binary output channel. Then there would be some 1f which could either change 11 to 01 or 01 to 11, and thereby cause an incorrect output to be produced over the channel. Hence the corresponding net could not be kd. Thus the above means that in any 2-wire, binary-output channel which comes for a kd net producing its f_i output over that channel, one wire of the channel must come from u merged realizations of f_i, $u > 0$, and the other wire must come from z merged realizations of f'_i, $z > 0$.

Now a net of the type asked for must be designed so that if 1 wire in one of its binary output channels is in the incorrect state, the other cannot be provided no more than a jf, $j < k$, occurs while the output was being determined. For there is always a 1f that can cause any output wire to go to the incorrect state. And if both wires are in the incorrect state, an incorrect output must be produced (assuming the results of the previous paragraph).

At this point, Theorem VIII can be concluded by using the results of the previous two paragraphs , the output characteristics of $k\mathcal{N}^*\mathcal{N}'^*$, an argument which is essentially the same as the one that was used to prove Lemma II, and the proof of Lemma V in that order.

Now, to move on to the case where $m > 2$, consider an m-wire, binary-output channel, C, which comes from a kd net, D, where D produces its f_i output over C. Let c1 and c2 be arbitrarily chosen bundle-state configurations of C which respectively represent correct (i. e., 0f) 0 and 1 outputs from D over C. Then, as in the proof of the previous Theorem, each wire in C can always go to an incorrect state in at least one direction for some 1f in D. However, it must not be possible for c1 to go to c2, or vice versa, as a result of a jf, $j < k$, in D.

Since generalizations on the level of those in Theorem VIII are hard to perceive for $m > 2$, let the following example be used to feel some of them out. Suppose m=6 and k=8 in D above. Then it might be that each of the first 3 of C's wires come from 2 merged realizations of f_i, and each of the last 3 of them came from 2 merged realizations of f'_i. If this is the case, c1=000111 and c2=111000 are satisfactory choices for c1 and c2 For then if f_i correctly equals 1, c1 cannot go to c2 for any jf, $j \nless 8$; and if f_i correctly equals 0, c2 cannot go to c1 for any jf, $j \nless 8$.

In order to discuss one of the alternative forms of D which is equally satisfactory, make the following definitions: Let $g_1, \ldots, g_6$ be binary functions of the n_i binary variables $x_{1_i}, \ldots, x_{n_i}$. Then define $g_i \cap g_j$ to be a binary function of these n_i variables such that its value is 1 only wherever both g_i and g_j are 1. Also define $g_i \cup g_j$ to be a binary function of the n_i variables such that its value is 1 only if either or both of g_i's and g_j's values are 1. Now let C's 1st wire come from 4 merged realizations of $f_i \cup g_1$, its 2nd wire come from 4 merged realizations of $f_i \cup g_2$, its 3rd wire come from 4 merged realizations of $f_i \cup g_3$, its 4th wire come from 4 merged realizations of $f'_1 \cup g_4$ its 5th wire come from 4 merged realizations of $f'_i \cup g_5$, and its 6th wire come from 4 merged realizations of $f'_i \cup g_6$. Further, let $g_i \cap g_i = 0$ for all $i \neq j$ and i, j=1, 2, 3. Also let $g_i \cap g_j = 0$ for all $i \neq j$ and i, j = 4, 5, 6. Finally, assume that $f_i \cap g_i = 0$ for j=1, 2, 3; $f'_i \cap g_j = 0$ for j = 4, 5, 6; $f_i \cup g_1 \cup g_2 \cup g_3 = 1$; $f'_i \cup g_4 \cup g_5 \cup g_6 = 1$ and that none of the g_i are zero. Then if f_i correctly equals 1 for some complete input to D[1] and if no jf's occur, $j > 0$, all of the $f_i \cup g_i$ wires are 1 and one of the $f'_i \cup g_i$ wires is 1. On the other hand, if $f'_i = 1$ instead, all of the $f'_i \cup g_i$ wires are 1 and one of the $f_i \cup g_i$ wires is 1. Now if f_i is correctly 1, no jf, $j \nless 8$ can cause both one of the $f'_i \cup g_i$ wires to go to 0 and two of the three $f_i \cup g_i$ wires to go to 0. Similarly for f'_i substituted for f_i in this statement. Thus a kd criterion for interpreting the outputs of C in this example is: if two of the $f_i \cup g_i$ wires are 0, $f_i = 0$; if two of the $f'_i \cup g_i$ wires are 0, $f_i = 1$; and otherwise the output is indeterminate. The advantage of

1. See the definition at the beginning of Chapter 2.

the scheme of this example over the first one above arises when f_i and f'_i are very costly to realize but most of the $f_i \cup g_i$ and $f'_i \cup g_i$ are not at all so.

The two examples just given furnish heuristic proof that a tremendous number of formal possibilities exist for $m > 2$; and that in general it is not at all apparent which possibility is most economical.

At this point, the essential features of the above discussion are formulated in more general terms. The formulation concerns the realization of most economical kd nets which must have all their individual outputs produced separately over m-wire channels. Let S_1 be the set of Of 0 configurations of C, S_2 the set of Of 1 configuration of C, and S_3 the set of indeterminate configurations of C. If D is most economical, these three sets are exhaustive. Then the wires of C must come from a sufficient number of merged and/or unmerged realizations of functions of the type $f_i \cup g_j$ and $f'_i \cup g_j$, for arbitrarily selected g_j's, so that no jf, $j < k$, can send a member of S_1 to a member of S_2, and vice versa. Thus, if f_i is correctly 1, at least enough $f_i \cup g_j$ type wires of C must remain 1 so that the interpretation of C's output configuration cannot be 0. Similarly with f'_i, f'_i g_j, 0 and 1 respectively. Now if $m > k$, then k+1 disjoint realizations of f_i (with one of the output wires split the required number of times) can be used instead of the $f_i \cup g_j$, $f'_i \cup g_j$ pairs scheme. The most economical kd realization of D is the one which is most economically realized in accord with the above set of conditions.

The next consideration is whether or not m-wire binary-output-channel DRL nets can be made kc for any greater k than non-DRL nets. Let the notation of the previous paragraphs be used. Then suppose that C had $[m/2]$ of its wires coming from merged and/or unmerged realizations of f_i and the other $m - [m/2]$ of its wires coming from merged and/or unmerged realizations of f'_i. Suppose further that the wires of C come from a sufficient number of merged realizations of f_i and f'_i so that the following is true: 1) if f_i is correctly 1, there are always fewer f_i than f'_i wires in the 0 state at least if no more than a ($\lceil m/2 \rceil$ -1) f occurs; and 2) if f_i is correctly 0, there are always fewer f'_i than f_i wires in the 0 state at least if no more than a ($\lceil m/2 \rceil$ -1) f occurs. Evidently 1) and 2) are possible without any merging at all. Now with the above provisions, D's f_i output over C is ($\lceil m/2 \rceil$ -1)c by regarding f_i as either 0 or 1, depending upon whether fewer f'_i or f_i wires of C respectively are 0.

Apparently there is no way of merging realizations into wires of C so that D's f_i output over C can be made jc, $j > (\lceil m/2 \rceil -1)$. For every merged output can always fail in 1 direction as a result of some 1f. Hence at best, a 1f can cause each of at least $\lceil m/2 \rceil$ wires in C to fail for at least one correct f_i state. And the remaining wires of C can be made to fail for a 1f when the other f_i state is the correct one. Thus under any dichotomy of wire-state-configuration representations of 1 and 0 in C, no more than all ($\lceil m/2 \rceil$ -1) f's can be corrected by C's output coding. But m disjoint realizations of f_i are ($\lceil m/2 \rceil$-1)c by Lemma I. Thus, <u>m-wire binary-output-channel DRL nets cannot be made kc for any greater k than non-DRL nets.</u>

<u>But DRL nets can be made kd as well as ($\lceil m/2 \rceil$ -1)c whereas non-DRL nets cannot.</u> The kd addition is accomplished by appending to 1) and 2) above the following: 3) if f_i is correctly 1, there are never more f_i than f'_i wires in the 0 state at least if no more than a kf occurs; and 4) if f_i is correctly 0, there are never more f'_i than f_i wires in the 0 state at least if no more than a kf occurs.

3) implies that if all the f'_i wires are incorrect, then none of the f_i wires can be incorrect. So each f_i wire in C must come from a merger of $k-(m-[m/2])-1$ realizations of f_i. Similarly, by 4), each f'_i wire in C must come from a merger of $k-[m/2]+1$ realizations of f'_i.

The final points of this Section follow: 1) By their nature, memory nets cannot be simply kd. Hence the study of the present Chapter cannot be extended in that direction; 2) Note that none of the kd DRL nets discussed in this Chapter have greater time delays than their irredundant counterparts; 3) time-domain redundancy can be used in DRL nets to obtain greater kd and kc characteristics just as was discussed in Chapter 2: that is, by using binary sequences for inputs instead of single 1's and 0's; 4) the input scheme used in this Section is justifiable to the extent that a) $k\mathcal{D}^*\mathcal{D}'^*$ nets produce outputs that either match it or are indeterminate, and 6) such inputs are usually about as easy to get as simple variable inputs, so because DRL nets are quite adapted to them they were used; and 5) Note that none of the results of this Chapter apply to the transmission of b bits, $b > 1$, over the same m-wire channel. All channels that were considered were binary.

CHAPTER 6

SOME FINAL REMARKS

Essentially, this study is concerned with nets that are composed of very reliable unilateral components. The fact that "and", "or", and "not" component-types were selected at the outset to base particular results upon is somewhat incidental. For the same general development would have followed from any sufficient set of unilateral components.

The major defect of the "k" theory that has been developed in this study is that it is of dubious value for relatively low component reliabilities, say e greater than .001. The thought behind using the k-degree failure criterion was to add more character to the extremely general problem of decreasing the probability of failure of logical nets. Unfortunately, only an extremely loose set of condition bounds could be obtained for matching the "k" criteria with the ideal probability criterion. Thus, the theory is only known to be completely relevant for components and net sizes which are currently in the range of engineering practicality.

It should be noted that information-theoretic or other criteria could just as well have been used to measure reliability as the "k" criteria. However none of them were because of the contra-intuitive nature of some of the first results they led to.

Although the theory was developed for synchronous nets, in principle it applies almost equally well to asynchronous ones. In fact only one change is required in the theory to make its whole development strictly applicable to asynchronous nets. This is that the set of admissible failure types be restricted so as to prevent nets' outputs from racing back and forth between 1 and 0 representations before settling.

An important feature of the net forms that have been developed in this study is that there are no "randomizing" subnets in them. Thus, wire bundle sizes are not required to be large enough to assure that their permutations will have good statistical properties. This allows more practical amounts of redundancy to be used in making given nets more reliable.

Finally, it should be re-emphasized that if majority organs had been available for use in the input-state resolver stages of 2k+1 nets, the principal cost of these nets would have been vastly reduced. In fact, it would have been reduced to little more than 2k+1 times the cost of the corresponding irredundant forms.

BIBLIOGRAPHY

BOOKS

(1) Cramer', Mathematical Methods of Statistics, Princeton Univer., 1951

(2) R. Richards, Arithmetic Operations in Digital Computers, Princeton, Van Nostrand, 1955

(3) Rosenbloom, Elements of Mathematical Logic, New York, Dover, 1950

(4) Rosser and Turquette, Many-Valued Logics, Amsterdam (Netherlands), North Holland, 1952

(5) Shannon and McCarthy (edit.), Automata Studies, Princeton Univer., 1956. Articles by Kleene, von Neuman, Minsky, Moore, DeLeeuw, et. al.

(6) Shannon and Weaver, The Mathematical Theory of Communication, Urbana, Univer. of Ill., 1949

(7) Staff, Harvard Computation Lab. (edit.), Forthcoming Book on the "International Symposium on the Theory of Switching" held at Harvard Univer. April 2, 3, 4, and 5, 1957.

PAPERS AND MONOGRAPHS

(8) Burks and Copi, "The Logical Design of an Idealized General Purpose Computer", Journal Franklin Institute, 261, 1956, pp. 299-314 and 421-436

(9) Burks and Wang, "The Logic of Automata", U. of Michigan Report under Air Force Contract No. AF18(603)-72, 1956.

(10) Ellis, "A Mathematical Formulation of the Generalized Logical Design Problem", 1957 IRE Wescon Convention Record, pp. 259-261

(11) Huffman, "The Synthesis of Sequential Switching Circuits", Journal Franklin Institute, March, April, 1954.

(12) McCluskey, "Minimization of Boolean Functions", Bell System Tech. Journal 35, Nov., 1956, pp. 1417-1444.

(13) Mealy, "A Method for the Synthesis of Sequential Switching Circuits" Bell Tele. Labs. Internal report.

(14) Moore and Shannon, "Reliable Circuits Using Less Reliable Relays", Bell System Monograph 2696.

(15) Robertson, J. E., "Error Detection and Correction in Binary Parallel Digital Computers. Electronic Digital Computer, Internal Report No. 37," U. of Ill. Digital Computer Lab., 1952.

(16) Roth, "Combinatorial Topological Methods in the Synthesis of Switching Circuits", IBM Research Report RC-11, 1957.

(17) C. Lee, "Several-Valued Combinational Switching Circuits", AIEE paper No. 56-162.

(18) Slepian, "A Class of Binary Signaling Alphabets", Bell System Monograph 2563.

(19) Schneider and Wagner, "Error Detection in Redundant Systems", Proc. of the Western Joint Computer Conference, (Feb. 1957, Los Angeles, Calif.). Conference Topic: Reliability.