Automatic trouble isolation in duplex central controls employing matching

by E. M. PRELL

Bell Telephone Laboratories, Incorporated

Holmdel, New Jersey

INTRODUCTION

At present in stored program control systems for switching telephone traffic, duplication is provided for dependability rather than to increase the traffic handling capacity of the system. Figure 1 depicts a simplified block diagram of the data processor portion of such a system.* The data processor is composed of central control and memory subsystems interconnected by communication buses. The memory subsystem** comprises two store groups each containing numerous information blocks (or stores). Each store of a particular store group has its corresponding image in the opposite store group. The stores contain two distinct categories of information: (1) semipermanent data (program and parameter), and (2) traffic variable data which are sometimes intermixed, but generally separated into different stores or even different memory subsystems. The data processor can be thought of as two central controls (CCs) which normally run in parallel, synchronously executing the same program, each from its own associated store group.

To be able to perform useful work in the switching system environment and for the system to be able to maintain itself, the CCs must also be able to receive information (scan) from points not normally accessible within their own environment and send information (distribute) to units outside of their own environment. One CC is defined to be active and the other standby.

Normally only the active CC controls the I/O equipments.

The data processor must be able to control I/O equipments which operate at speeds incompatible with its basic processing speed. This requires that the system be multiprogrammed. Also, it is more efficient in a multiprogrammed system to employ interrupts to allow each program subsystem to operate at a rate appropriate for the equipment and/or processing function involved. For example, one interrupt level allows I/O programs to be executed at rates dictated by the I/O equipments. Higher interrupt levels allow maintenance programs to make high priority claims upon the data processor when trouble occurs.

Duplication and switching are necessary to allow the system to operate in the presence of hardware troubles. Thus, for example, the system must be able to operate with a store out in store group 0 and a central control out in data processor 1, see Figure 1. Generally, all communication buses are completely switchable at the central control; i.e., either or both buses can be connected to either or both central controls. Hence, various central control-communication bus modes can be established by means of unique configurations of the buses and central controls.

These systems depend primarily on hardware checking circuits for trouble detection during data processing. When a trouble detection circuit detects a trouble in the system, it notifies an “interrupt” circuit. The interrupt circuit immediately stops operational program processing and transfers control to a fault recognition (FOR) program associated with the particular type of trouble indication. The functions of the FOR programs are to determine quickly an
operational system configuration, establish it by switching out faulty units, and then return to operational program processing. Sometimes, the ability of these programs to perform their function is impaired by the fault in the system. When this occurs, a hardware back-up system in conjunction with a checking program, control the data processor reconfiguration. This facility is designed as an autonomous circuit within the central controls and is initiated by various timing circuits.

![Diagram](image)

Figure 1 – The data processor

Thus, the FOR programs are basically tests of processing integrity (sanity) of the system. These programs, with the help of additional hardware features, minimize the probability of an inoperative configuration mutilating temporary information. To minimize the amount of time taken from operational program processing, these programs generally try to use brief and direct methods to discriminate between error conditions (non-repeatable) and repeatable troubles (faults). When a fault is detected, sufficient testing of the duplicate units is performed to find one complete working system, and after doing so, to return to operational program processing.

Fault Recognition is the most important maintenance function in a real-time system. The minimization of time taken from the processing function with an extremely high probability of returning to processing with an operational (sane) system is the critical objective which any automated maintenance technique must satisfy. Diagnostics with "fine" resolution are of secondary importance.

The FOR programs will request that each unit found faulty or suspected of being faulty be diagnosed. The function of the diagnostic programs (which are interleaved with operational program processing) is to generate test data to isolate the fault to a reasonably small section within the unit. Maintenance dictionaries are provided for translating these test results to the location of the faulty section. By using standardized packages and plug-in techniques, faulty components can be readily removed from the system.

The central control is the most difficult part of the system for which to design maintenance programs. This is because of its critical position in the structure, its size, and its lack of a high degree of symmetry. This paper concerns itself with two basic methods developed at Bell Telephone Laboratories to cope with this problem.

Central control maintenance – general considerations

During operational program processing as mentioned earlier, the central controls are normally synchronously executing the same program. In addition, a circuit between* the central controls is used to inform each central control of the operations being performed by its mate. This circuit is called a match circuit. It provides programmatic access to the microsteps within the complex central control operations. A simple way of looking at the match circuits is as follows: central controls 0 and 1 add \(1 + 1 = 2\). Then each central control with its own** match circuit verifies that the same inputs and outputs occurred with the adders of the two central controls. When either central control detects a mismatch, a FOR program is entered.

The first purpose of FOR programs is to filter out random errors. If the match circuits are fast enough, and the machine language structure clean enough, the FOR programs can actually unwind the instruction (or instructions) so that a retrial of the failed operation can be attempted. Unwinding generally refers to reestablishing the state at the beginning of a partially executed instruction or of a nested sequence. A second failure usually implies a faulty unit. Otherwise, an error occurred and both machines are returned to the interrupted operational program.

However, in some machines the logical complexity is such that they cannot be stopped on mismatches

---

*Functionally only.

**In some machines one central control may look at inputs at one moment in time and the other at outputs.
Automatic Trouble Isolation In Duplex Central Controls Employing Matching

before program accessible data in the machine has been mutilated. In this case, program unwinding is impossible. Hence, what the matcher (or matchers) match then must be a function of the operation (or operations) in progress within the central control(s).

In this case, the FOR program actually attempts to reproduce the trouble symptom under controlled conditions by logically exercising the central control hardware which most likely caused the interrupt. (This exercise program is a subset of the complete FOR program.) If no trouble symptom can be reproduced, the problem is classified as an error for subsequent analysis and the machines return to operational program processing. In either case these programs lead to a complete check of all central control hardware if a fault symptom is detected.

There are two basic techniques for fault localization. The first method makes use of a decision tree where at each node within the tree, the fault space is partitioned into two or more hopefully disjoint and almost equal smaller possible fault spaces. Resultingly, the tree leads to a set of terminal nodes each of which determines a few possible faults in the unit. Because of the difficulty of designing a decision tree for relatively large complex circuits such as a central control, a method is sought based on executing a more or less fixed set of functional tests which produce an output derived from the failing tests. This output would then be program translated into a number which would identify the faulty circuit pack(s) in a printed dictionary which would be compiled by fault simulation. This paper considers diagnostic test design basically from the latter point of view. In addition, the difficulties which arise in attempting to consider the entire unit under diagnosis at the time of test design led naturally to the concept of dividing the unit into functional circuit blocks, and then considering each block individually. In doing this, the program designer, when testing block i, assumes that the fault is in block i. Thus, the other circuit blocks can be used as test tools in testing block i. But the division of the program into functional blocks of tests (called phases) creates relatively complex interfaces between these blocks as well as creating many problems relating to consistency of the test results.

To minimize the "hard-core" of circuits which must function to properly execute the FOR program, much thought must go into the functional division and ordering of the functional blocks. The functional division and ordering is even more important in diagnostics due to inconsistency problems. The objective is to use only tested circuits to test other circuits.

Distinct central control fault recognition and diagnostic programs

A. Introduction

In the No. 1 ESS stored program switching system developed at Bell Telephone Laboratories, the FOR and diagnostic (DIAG) programs are separate entities. The basic strategies of the separate FOR and DIAG programs are outlined in this section. These strategies are largely carried over to the integrated program approach discussed in the next section. The differences in the two approaches lie primarily in program structure and the use of the matching facilities.

B. Fault recognition

The basic program flow technique used within test phases in FOR programs is shown on Figure 2. The tests are designed as if each central control is testing itself under the assumption of no redundant logic such that a single failure would go undetected. "Tests consist of data manipulation operations followed by conditional transfer orders which check for the proper circuit response. If the active central control fails, it will attempt to switch central controls (which it alone can do by program). The faulty unit, now standby, will be removed from service and the diagnostic program requested. If the active passes, checks are made to see if the standby is still in step (by examining the match circuits). If the standby is found to be out of step, it is removed from service, since it failed to follow the operations of the good active unit. This testing process is continued until all tests have passed or a faulty unit is found."

C. Diagnostic

In the DIAG program, it is known that the standby is faulty and that the active is fault-free. Thus, the active can be used to test the standby. The assumptions of single failure and no redundant logic are applied when designing tests. Again, the test programs are broken into phases. The execution of a test follows a strict pattern as follows:

1Excessive error rates can lead to complete checks and diagnosis of both central controls. If the error rates do not subside, teletype writer message(s) will follow indicating the inability of the programs to isolate the reason for the high error rates.

2Distinct central control fault recognition and diagnostic programs

A. Introduction

In the No. 1 ESS stored program switching system developed at Bell Telephone Laboratories, the FOR and diagnostic (DIAG) programs are separate entities. The basic strategies of the separate FOR and DIAG programs are outlined in this section. These strategies are largely carried over to the integrated program approach discussed in the next section. The differences in the two approaches lie primarily in program structure and the use of the matching facilities.

B. Fault recognition

The basic program flow technique used within test phases in FOR programs is shown on Figure 2. The tests are designed as if each central control is testing itself under the assumption of no redundant logic such that a single failure would go undetected. "Tests consist of data manipulation operations followed by conditional transfer orders which check for the proper circuit response. If the active central control fails, it will attempt to switch central controls (which it alone can do by program). The faulty unit, now standby, will be removed from service and the diagnostic program requested. If the active passes, checks are made to see if the standby is still in step (by examining the match circuits). If the standby is found to be out of step, it is removed from service, since it failed to follow the operations of the good active unit. This testing process is continued until all tests have passed or a faulty unit is found."

C. Diagnostic

In the DIAG program, it is known that the standby is faulty and that the active is fault-free. Thus, the active can be used to test the standby. The assumptions of single failure and no redundant logic are applied when designing tests. Again, the test programs are broken into phases. The execution of a test follows a strict pattern as follows:

1By inconsistency it is meant that when a given fault is inserted during fault simulation, the diagnostic program does not produce the same set of failing tests with repeated diagnosis or that the set of failing tests produced during laboratory fault simulation is not identical to that produced when the same (as the simulated) fault occurs in the field.
The integrated approach to central control fault recognition and diagnostics

A. Introduction

"Combined" and "Integrated" are used to define a single program which is used for both FOR and DIAG. "Combined" denotes a program which simultaneously performs FOR and DIAG. This is the method usually employed in commercial computers.\textsuperscript{4,5} This approach cannot be used in systems which operate in real time because the execution time of a program which records diagnostic data from and through a match system is at least an order of magnitude greater than that of the corresponding program without data recording.\textsuperscript{**} This type of increase in FOR execution time from say 25 MS to 250 MS cannot be tolerated in most real-time systems, and specifically not in the No. 1 ESS.

"Integrated" denotes a program whose course of action is dependent upon its use; i.e., FOR or DIAG. This method is based on the assumption that with sufficient match access, FOR is a subset of DIAG; that is, those procedures which test the logical capability of a central control also provide results for diagnostic resolution. For example, the logical tests used to test an adder during fault recognition would also be employed during diagnosis except that in diagnosis the matchers would be used to look at the input and output of the adder.

B. Matching features essential to this method

The primary means of trouble detection for the central controls is the matchers. They do, however, have a secondary function of providing a means for diagnosis of the central controls; i.e., they provide access to many points within the central controls. Since the match system does play a major role in the maintenance scheme, it should be intrinsically reliable. This means that the failure rate of the match circuitry should be small in comparison to the failure rate of the central control. To this end, it is imperative that the matchers be designed mainly for adequate trouble detection. Next, a judicious choice of the requirements for system diagnostic capability should be made if the matchers are not to be overly complex and become a burden on the system.

Basically the match system must be able to perform both a directed match and a sampled match. In a directed match, the matchers are directed to look

\textsuperscript{*}Some of the reasons for this are that in DIAG the match sources and time segments (see section B) must be continuously changed to obtain the desired diagnostic data, whereas, in FOR the sources and time segment(s) are fixed throughout execution. In addition, in DIAG each failure or even each test requires data recording, matcher reinitialization, truncation decisions to avoid inconsistency, etc.
at match sources at specified time segments on a continuous basis. In the sampled match, the matchers are directed to look at specified sources at a specified time segment a given number of machine cycles from the point of initialization. This is frequently called selective, discrete, or snapshot matching.

The No. 1 EES relied heavily on discrete sampling at predetermined points within a manually written test program. This is a powerful tool and it is used in certain DIAG only tests in the integrated program method.

However, the integrated tests use the directed match mode with flexibility provided by varying the match sources and/or the segments of the machine cycle when matching occurs. In this mode the specified sources are matched continuously on every machine cycle at one or more of the three time segments. This is the same mode that is used during operational program processing, but then the sources are set to the internal communication buses within the central controls. In diagnosis a special interrupt facility is initiated by the matcher(s) to call in a diagnostic recording routine when an abnormality is detected. In addition, the matcher will indicate the number of machine cycles or instructions since the last interrupt. This indication is referred to as a timeout count (TOC). The special interrupt facility can also be used with discrete sampling.

C. Integrated approach

In this approach a single functional block (or phase) of tests performs both diagnostic and fault recognition functions. The basic difference between using the program for FOR or DIAG is the collection of data. When this program is run as a FOR program, data about the standby central control is not collected. Whereas, when the program is run as a diagnostic, a failure in test i will result in the saving of the TOC and the bit pattern of mismatch. A general flowchart for the integrated program for test i is shown on Figure 3. Note that test i now includes all instructions including initialization. Thus, this is more of a continuous sampling technique. In addition, the testing is continued regardless of failures and further data is gathered.

* A variation because of machine complexity is for the matcher(s) to be directed to look at a set of match sources on a sequential basis. The source to be matched may also become a function of the instruction(s) being processed.

†† The matcher can be directed to interrupt upon detecting either a match or mismatch between the two machines.

* When executing the program for diagnosis, there exists the possibility of not executing phases that passed when the program was executed for fault recognition. This is accomplished if a minimum amount of recording is performed during fault recognition.

With this approach the part of the FOR program in the previously described approach used to completely test for faults is essentially eliminated. This results in approximately a 20 percent reduction in program size. However, in the integrated approach, there is a tendency towards some increase in recovery time for faults since more comprehensive diagnostic type test sequences tend to be used during fault recognition. It is also true that having separate programs allows more freedom in modifying the FOR program.

Further work

This section describes some of the more interesting ideas being considered for further improvements in these automated maintenance programs.

Two important factors in developing any automatic diagnostic technique are design time and memory cost. Both are large. Program sizes in terms of program words for diagnostics are approximately equal to the number of transistors in the unit. At present, the design of a set of sufficient tests has been obtained to a large extent by manual analysis and revised as indicated by empirical evaluation. Work has been going on for some time on automatic generation of a set of tests and the methods have been applied successfully to limited size circuit blocks. Prospects for applying these techniques to large combinational circuits look promising.
Memory costs will most likely be reduced in the future by storing diagnostic programs in a cheaper medium such as magnetic disk (or tape) and bringing them into temporary memory as required.

More work needs to be done in detecting and distinguishing different faults with a consistent pattern. Adequate distinguishability or resolution has not been too difficult to obtain, but detectability is only at about 80 percent of the set of faults considered. In designing a test for a particular class of faults, the reaction of this test to other possibly unrelated faults affects consistency. Possibly a combination of the decision-tree and failure pattern technique can bring about better consistency. The decision-tree would be used to partition the initial fault space into subsets of say a 100 or more possible faults and then the failure pattern technique could be used for further resolution.

Another possible avenue is the cell dictionary. At present an exact match of the output number derived from the diagnostic results must be found in the dictionary of fault numbers generated by fault simulation. The cell dictionary would assign all faults to regions or cells of the diagnostic data space. If an exact match could not be found, the diagnostic cell dictionary output would list the closest cell centers to the actual diagnostic data pattern. This is only one of many other dictionary techniques being explored. Early results look promising that one or more of these techniques will minimize the consistency problem.

Further improvements in mechanisms for data gathering to reduce the hard-core of circuits can be developed with an associated smaller matching system. The direct use of memory by the standby processor with the active processor sending store control signals looks promising here.

The present approach to self-diagnosis makes design changes difficult since the dictionaries usually must be regenerated. It would be highly desirable for small changes in circuit logic to be reflected in small changes in diagnostic programs and the associated dictionaries. Intermittent troubles are also a problem which are presently being solved by automatic error analysis followed by repeated diagnosis and off-line testing by highly trained maintenance personnel. How to function with less trained maintenance personnel who will have limited troubleshooting experience due to the high reliability of electronic circuits remains a problem.

For the future, integrated circuits are going to require a whole new set of maintenance ground rules and revised automatic maintenance strategy and program. Hopefully, the work painstakingly done with the present type of circuits will make this task easier.

ACKNOWLEDGMENTS

The author is especially indebted to Messrs. R. W. Downing, W. C. Lehrman, and J. M. Lix for their valuable assistance during the development of the integrated program.

REFERENCES

1. R W DOWNING J S NOWAK L S TUOMENOKSA
   No J ESS maintenance plan
   Bell System Technical Journal part I September 1964
2. R W DOWNING J S NOWAK L S TUOMENOKSA
   Maintenance planning on no J ESS
   IEEE Annual Communications Convention Colorado June 1965
3. R L CAMPBELL
   General maintenance techniques for large digital systems
   5th Annual Reliability Maintainability Conference N Y C July 1966
4. K MALING E L ALLEN
   A computer organization and programming system for automated maintenance
   IEEE Transactions on Electronic Computers December 1963
5. R V BOCK A P TOTH
   Hardware and software for maintenance in the B5500 processor
   IEEE International Convention N Y C March 1965
6. H Y CHANG
   Methods of interpreting diagnostic data for locating faults in digital machines
   IEEE Design Automation Workshop Michigan State U September 1966
7. H Y CHANG W THOMIS
   Methods of interpreting diagnostic data for locating faults in digital machines
   Bell System Technical Journal to be published