INTERDEPARTMENTAL

MASSACHUSETTS INSTITUTE OF TECHNOLOGY *CAMBRIDGE 39, MASS.*

from the office of E. Manning

January 20, 1966

TO: Professors Corbato
Dennis
Glaser
Saltzer

FROM: Eric Manning

SUBJECT: 645 Diagnosis Plan

The enclosed is a first draft of material to be discussed at the IEEE Reliable Automata Workshop on February 2. I would therfore appreciate your criticisms during the early part of next week, if at all possible.

Thanks,

Eric Manning

A PRELIMINARY SELF-DIAGNOSIS PLAN FOR THE GE-645

Eric Manning

This note is the result of a first look at the GE-645 from the viewpoint of self-diagnosis. It is intended to assist the GE-Phoenix Self-Diagnosis Group, which will put the proposed method to the experimental test, and to acquaint other interested people with progress to date. It describes a basic strategy for attacking the problem, and attempts to show how the structure of the 645 will permit a useful solution. A step-by-step routine for generating the self-diagnosis procedure is outlined. Finally, obvious areas for further work are listed.

The references given are all relevant, and the reader should in particular be familiar with [1].

1) The Problem:

A self-diagnosis procedure is an experiment of exactly two outcomes:

1) All transistor-diode logic is failure-free

or 2) Card or module X has a fault of type y.

We make the following assumptions:

1) The class of failures is finite and known.
   (Specifically, we allow open diodes, and transistor gate outputs stuck at logical 0 or 1)
2) Every failure transforms the sequential circuit into another sequential circuit.

3) Failures are not intermittent.

4) Feedback lines can be momentarily reset, even under failure, to a known initial state.

5) At most one failure has occurred since the last diagnosis [the crucial single-fault assumption].

Further, we only consider failures of Central Processors (CPs) and memory controllers (MCs) in this note. In addition, we orient our effort towards checkout, rather than diagnosis. [Fast detection of failures is, of course, essential. Also, it often turns out that a checkout test sequence provides sufficient diagnostic information]. Finally, we do not attempt to intergrate the procedure with the MULTICS System.

2) Properties of a Solution:

1) The procedure should be fast enough to allow execution every few hours, in order to validate the single-fault assumption.

2) The amount of digital simulation required to generate the procedure must not be prohibitive. [It is felt that this requirement poses the major obstacle to treatment of a large system].

3) It must be possible for the GE-645 to provide service to users while testing is in progress. (on-the-fly diagnosis).

4) The procedure should be general. That is, little or no additional simulation should be required to adapt it to other GE-645 configurations.

3) Review:

We briefly restate some concepts from Reference [1]. (However, a careful reading of [1] is still prerequisite to a thorough grasp of this note).

a) The Simulation Program

The basic tool of our approach to self-diagnosis is a version of the Sequential Analyzer programming system, developed by the late Professor S. Seshu. This system accepts a coded description of a sequential circuit as input, and compiles a simulator which simulates the behavior of both the good circuit, and of the class of failed circuits generated therefrom. The reset feedback state and sequence of primary inputs to be applied to the class of circuits is supplied by the user. Application of the reset state imposes a partition with respect to output vector on the class of machines. This particular, together with the output vectors of the partition-blocks, is reported as the output from the program. Each input vector subsequently applied causes a refinement of the

partition, and all refinements are reported as Simulation proceeds. A checkout sequence, then, is simply an input sequence yielding a final partition such that the good circuit is in a partition-block by itself.

Definition: The computer on which the simulation program is run is the production machine. (Here, a GE-635).

Definition: The computer for which the self-diagnosis procedure is being produced is the subject machine. (The GE-645).

The simulation program allows us to determine the effect of every machine failure of the subject on any instruction of the subject's order code. To do so, we choose a piece of the subject machine as the circuit to be simulated. The circuit input lines are taken as the directed outset from the remainder of the subject to the piece chosen. The output lines can be chosen arbitrarily, but the observable live registers are usually selected.

A sequence of input vectors which correspond to the execution of a machine instruction is made up, and is read by the simulation program. The output of the program gives the desired result.

The machines with failure may behave in any of several ways. A failed machine may reach a Stable State, and fail to complete the instruction sequence. The output vector is interpreted to identify this case, and to provide the information necessary to associate the state reached with a particular subset of failures. Or, the failed machine may complete all instructions, but produce a wrong answer in one of the observable registers. This case is also identified by interpretation of output vectors, and is treated by appending a string of instructions leading to a DIS (conversion string), or to a message to the operator. The conversion or message-producing string is simulated on the subset of machines in question, to ensure that the corresponding failures do not incapacitate the instructions of the string.

A third possibility is that a failure-induced oscillation or critical race may occur. The present simulation program identifies such machines, but simply discards them. Finally, the failed-machine behavior may be everywhere identical to that of the good machine. In this case, a redundant component, which should be removed, has failed.

Thus, it is possible to select a sequence of machine instructions which will run to completion if and only if the subject machine is good. We note that this is a self-diagnosis procedure in the sense that a failed processor detects its own faults - a second processor is not required to test it.

4) Strategy For a Multi-Processor System:

The obvious and central issue raised by a multi-processor system is that of external versus self-diagnosis. External diagnosis [whereby one processor-memory controller combination or subcomputer tests another] is a well-known technique, and has been applied by several workers [2,3]. Self-diagnosis [whereby a subcomputer tests itself] is a more novel approach, and has not yet been widely tested. We outlined characteristics of both methods.

a) Characteristics of External Diagnosis

i) It is the more straight forward approach.

ii) It can be implemented on the GE-645 through the use of shared core, and the DIS, XED, and CON orders.

iii) Due to the single-fault assumption, there is always at least one good subcomputer available to diagnose faults of a failed subcomputer. The response of the failed subcomputer under test is generally observable, in that the SREG and STC1 instructions can be used to store the live register contents in core, from where the testing subcomputer may retrieve them. (We make the usual assumption that any failure of a non-redundant component will eventually result in wrong contents of one or more of these.)

iv) The production cost is twice that of self-diagnosis. For, it is necessary to simulate both the case
"Subcomputer A tests Subcomputer B and the fault is in B";
and the case
"Subcomputer A tests subcomputer B and the fault is in A".

v) Two subcomputers (i.e. both CPs and two MCs of the MAC machine) must be allocated for diagnosis. Thus, concurrent diagnosis and service to users will be impossible.

b) Characteristics of Self-Diagnosis

i) The method has not been used widely.

ii) The excellent logical and topological modularity of the 645 facilitate implementation of the method. Specifically, it is easy to isolate and treat a single subcomputer by shutting down memory ports. Also, the effects of a failure will be localized to the failed module.

iii) There may exist certain processor failures which block

attempts by the failed processor to detect them. For example, attempts to detect failures of logic used by the COMPARE instruction by comparing accumulator contents to stored words may fail. However, experience has indicated that such faults will comprise only a small percentage of the total.

iv) Simulation cost should be about half of that for the external-diagnosis method, as the two cases outlined in a) iv) do not arise.

v) As a single subcomputer tests itself, concurrent service to users of a system of two or more processors is possible.

vi) Resolution of faults may not be so good. For example, most faults causing a "dead" subcomputer will be indistinguishable, unless manual tests are employed.

c) The Proposed Method

We propose a mixture of self and external diagnosis, in an attempt to realize advantages of both approaches. To treat processors, we plan to check each processor as far as possible by self-diagnosis. If any processor failures are found which block their own detection, external diagnosis will be used to cope with them. The analogous technique for memory controllers is as follows.

It is expected that a large fraction of memory controller faults will cause a "dead" subcomputer. We therefore plan to perform checkout on all possible subcomputers of the system, to improve resolution. Thus, an outcome such as:

$CP_1 \cup MC_1$ dead

$CP_1 \cup MC_2$ checked out

clearly permits resolution of the fault to MC_1. We may note that the ability to form many subcomputers, inherent in the 645 architecture, is of benefit here.

To summarize, it appears that the possibility of undetectable faults of nonredundant components is removed by the proposed method. The production cost will be about the same as for self-diagnosis. Uninterrupted service to users will be possible, except during the (relatively short) phase

of processor external diagnosis. (However, note that the external diagnosis phase can be delayed until a period of low user activity, as long as the memory traces are preserved.) Finally, failure resolution is expected to be somewhat better than that of pure self-diagnosis.

5) Outline of the Mixed Procedure

To initiate checkout, the operator shuts down certain memory ports by console switches, thus bisecting the system into subcomputers S_1 and S_2 where

$$S_1 = CP_1 \cup MC_1 \cup MC_2$$
$$S_2 = CP_2 \cup MC_3 \cup MC_4$$ for the MAC GE-645.

He then attempts to reset the subcomputer under test, say S_1, and load the diagnostic. The checkout sequences for CP_1 and MC_1 are executed. Now MC_2 is called (automatically, by reloading the program into a different address range), and the sequences are repeated. Execution is terminated by a DIS (delay until interrupt) instruction.

The operator enables the $CP_2 \longleftrightarrow MC_2$ memory port to initiate the external diagnosis phase. CP_2 is used to examine computed results left in memory by CP_1. After external diagnosis is complete, the operator repeats the process, with the roles of S_1 and S_2 exchanged.

6) The Production of the Procedure:

Due to the system architecture of the GE-645, it is only necessary to treat one subcomputer. This is most fortunate, as it would otherwise be impossible to produce a self-diagnosis procedure, due to the enormous amount of simulation required. The same fact also implies that the diagnostic may be applied to larger GE-645 configurations, with little or no additional simulation. We now outline the steps to be followed to produce a procedure, using the (a modified version of) simulation program described in [1]. Familiarity with reference [1] is assumed for this discussion.

a) Preparation of Input

The coded description of the logic of one CP and one MC is extracted from the GE design file. The input vector consists of the global feedback lines which must be broken to allow simulation in "steps" of one machine instruction; (the lines from core to the MC are driven by a memory interpretive routine which supplies data words to the MC), and the interface lines from the GLOC to the MC (one standard memory controller interface).

The output vector consists of one memory controller interface, lines to core, and the bits of all observable machine line registers. The p-subvector, of course, consists of

i) control lines whose misbehavior can cause hangup,

ii) the observable line registers of the CP.

b) Failure Classes

The set of subcomputer failures will be divided into failure-classes in the obvious way. Each failure-class will be a subset of failures

on either the CP or the MC. The size and number of failure-classes will be determined by the amount of core memory available on the GE-635 used for simulation. The use of disc or drum storage is to be avoided, to reduce simulation time.

c) Simulation Runs

i) Failures of the first failure-calss of CP failures are compiled.

ii) An input-vector sequence corresponding to execution of the following actions is prepared.

a) Reset the subcomputer
b) Execute bootstrap loader, to load the checkout sequence
c) Execute the CP checkout sequence

[The CP checkout sequence can be generated fairly easily from the micro-order structure of the machine orders. Essentially, each new order is chosen to exercise as many new micro-orders as possible. After each instruction we insert a SREG, STC1 sequence to store observable registers for later testing.]

iii) A sequence of "load, compare, DIS if unequal" strings is executed to examine stored results, causing DIS to be executed in the case of failure.

iv) Each failed-machine subset is picked up and carried until either a hangup or a DIS occurs.

v) Steps i) - iv) are repeated for other failure-classes, extending the checkout sequence as necessary.

vi) Steps i) - v) are repeated using different regions of memory, to exercise the other MCs.

vii) We have now finished the simulation of the self-diagnosis phase. We list all undetected failures, which are to be treated by external diagnosis. The care of a good testing computer can be handled manually. To treat the other case, we recompile all faults and simulate the external diagnosis sequence. This sequence will, of course, be of the "load, compare, DIS if unequal" variety.

viii) Finally, we perform local cross-check as discussed in [1]. A simple tape-sorting routine can be used here.

ix) The operator's manual, indexed by program counter contents at hangup or DIS, and specifying cards to replace, can now be made up (automatically if desired).

6) Summary

We summarize several points for convenience.

a) The machine

The well-modularized, multi-processor design of the GE-645 allows the physical isolation of subcomputers, and allows us to cope with the entire system by treating one subcomputer. This is probably necessary for diagnosis of such a large system to be feasible. The same fact allows user service by other subcomputers while diagnosis is in progress.

On the other hand, submodules of the GE-645 are internally clocked, and thus liable to timing failures. For the present we ignore these, as no adequate method of enumerating them is known to the author. This is done by ignoring clocked feedback variables during Huffman Analysis.

b) The method

We compare the features with the previously stated specification.

Execution time will be reasonable, as the checkout sequence for one subcomputer will be of the order of a few hundred instructions, thus a few milliseconds execution time.

Whether or not the amount of digital simulation required is feasible remains to be seen. However, the proposed method is perhaps optimal in the sense that it is hard to see how to treat the system by simulating anything less than one CP and one memory control.

Concurrent user service will be possible during the bulk of the procedure. The external diagnosis phase can be delayed until an opportune moment.

The procedure is certainly general, in that it is simply repeated tc treat for additional subcomputers.

Consequently, it would seem that the proposed method meets the specification.

8) Further Work

The procedure must be extended to treat input-output controllers, and disc and drum controllers. Treatment of the input-output controllers is being deferred until the author has a better grasp of their operating environment. The disc and drum controllers will probably be treated in a manner analogous to that chosen for memory controllers.

Another obvious extension is the development of diagnostic (as opposed to checkout) sequences, to improve fault resolution. For example, once it is established that a certain memory controller has failed, a good subcomputer could be used to test it further to obtain more diagnostic information. This possibility will not be explored until enough experimental results have been obtained to indicate if it is necessary.

Finally, it will eventually be necessary tominimize degradation of user service caused by diagnosis. To do this, it may be possible to divide the procedure into subprocedures, which can be interleaved with normal user service. This would require that the state information about the subcomputer under test be stored at the end of each subprocedure. It would also be necessary to make the procedures compatible with the MULTICS operating system.

References

1. Manning, E. G., "Self-Diagnosis of Electronic Computers - An Experimental Study," Coordinated Science Lab. Report R-259, University of Illinois, Urbana, Illinois, July 1965.

2. Tsiang, S. H. and W. Ulrich, "Automatic Trouble Diagnosis of Complex Logic Circuits," Bell System Technical Journal, Vol. 41, pp. 1177-1200, July 1962.

3. Forbes, D. E. et. al., "A Self-Diagnosible Computer," Proceedings of the F.J.C.C., Vol. 27, pp. 1073-1086, November 1965.