THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 53, No. 8, October 1974
Printed in U.S.A.

LAMP:

Application to Switching-System Development

By T. T. BUTLER, T. G. HALLIN, J. J. KULZER, and K. W. JOHNSON

(Manuscript received March 26, 1974)

Specific attempts have been made at Bell Laboratories to shorten development intervals, to improve the quality of system design, and to improve unit, manufacturing, and system testing by widespread application of the LAMP system during development of the 1A Processor and No. 4 Electronic Switching System. LAMP has played a major role in two areas of the design process—design verification and fault simulation. Although ten major digital subsystems of No. 4 ESS and the 1A Processor are now being simulated on the LAMP system, this paper describes the experience gained during development of two of the 1A Processor subsystems, central control and program/call store.

I. INTRODUCTION

Computerized development aids have become an integral part of the development of large, complex, electronic systems. One such aid, the LAMP system, is finding widespread application in the development of the 1A Processor,[1] a stored program processor, and in No. 4 ESS,[2] a new switching system that employs a solid-state, time-division, digital-switching network.

The need for computerized aids in the development of advanced switching systems is vital for several reasons. Vast amounts of engineering and manufacturing information must be generated. Complicated design decisions coming from engineers of diverse disciplines must be coordinated, and with the increasing complexity of systems and electronic technology comes the need for more thorough and consistent testing at all stages of design and manufacture. As the computer increases in power, it plays a greater role in reducing manual design effort, enhancing design quality, improving the accuracy of information transfer, and making more complex designs economically feasible.

For these reasons, the LAMP system has been used on TSS 360/67 to simulate ten major subsystems in No. 4 ESS and the 1A Processor. The experience gained during the development of two of these subsystems, central control (the heart of the 1A Processor, which provides program execution and overall executive control) and program/call store (a magnetic core memory unit that provides storage for ESS programs and temporary call-related data) is detailed here.

To facilitate maintenance of design information and provision of adequate testing for the project, computerized data bases have been implemented for all design information. The combination of common data bases for hardware and software design information, the LAMP simulator, and the conversational features of an interactive host computer have proven quite effective in the hardware and test design for the project.

There are two major ways in which the LAMP system has been used in the development process: design verification and fault simulation. These will be discussed separately and in detail in Sections II and III, respectively, of this paper. Briefly, design verification consists of demonstrating that the unit being simulated performs the functions it was designed to perform, with no faults present. Fault simulation, on the other hand, consists of inserting faults into the simulated unit and testing the ability of maintenance programs to detect and isolate those faults.

Three categories of test programs are used in the simulation of the previously mentioned units. They are:

(*i*) Circuit pack level tests, which are used in design verification, fault simulation, and pack testing.

(*ii*) Diagnostic tests, which are the primary tool for both design verification and fault simulation at the complete unit level. The diagnostic tests are written in a high-level language, concurrent with the design of the hardware. They are intended for factory and installation tests as well as for on-line fault detection and repair in an operating system. Total fault detection is the ideal primary goal, with good resolution the secondary goal.

(*iii*) "Special" test programs, which are used only for design verification and are intended to test the functional capability of the simulated unit (e.g., its ability to execute the program), and to test complex interactions between different portions of the unit.

Initially, LAMP was used to simulate each digital circuit pack of the 1A Processor (*i*) to verify the design, (*ii*) to verify that sufficient

access was available on the pack to detect all classical faults (output stuck high or low, open input), and (*iii*) to verify that the tests were capable of detecting these faults. This circuit pack level of simulation continued throughout the development process as changes were made to circuit packs and new packs were issued.

A second, temporary phase for some units was the simulation of a functional part of a unit before the complete design was available. This allowed design verification through simulation to begin while other functions were being designed. At this level, many special test programs described previously were used.

Finally, as the complete design became available, complete unit simulation was begun. At this level, the diagnostic tests were used, and the majority of design verification and fault simulation was done.

II. DESIGN VERIFICATION

2.1 Circuit pack design verification

This section describes the use of LAMP simulation in the design verification of 1A Processor circuit packs. This is differentiated from design verification at the complete unit level and from fault simulation of circuit packs, which will be discussed in later sections.

2.1.1 Objectives

Substantial time and expense are required to produce an artmaster for a 1A Processor circuit pack (100 to 400 logic gates) and then to produce the first hardware version of the pack. It becomes important, therefore, to verify the accuracy of the design before this process begins. The purpose is to test the ability of the design to perform the intended functions as completely as possible. In addition to new designs for circuit packs, changes inevitably must be made during the course of the project. Again, it is important that the design of these changes be verified before the time-consuming process of modifying the hardware begins. For this reason, design verification of circuit packs, through simulation, is done not only early in the development process but throughout its course.

2.1.2 Circuit pack simulation

The mechanics of circuit pack design verification by simulation consist of building a LAMP model of the pack, devising a set of tests, running the tests and interpreting the results, and updating the model so that proposed corrections of design errors may be tested immediately. Building and updating the model are described under complete unit simulation (Section 2.2.2).

2.1.3 Circuit pack tests

Design verification at the circuit pack level uses tests written by the pack designer or another person familiar with the design and information generated automatically via the automatic test generation (ATG) program.[3] The handwritten tests consist of sets of inputs (vectors) to be applied to the circuit pack. During simulation of the vectors, the outputs of the pack are continuously monitored and, thus, may be compared to a set of expected results. An effort is made to make a complete "active" test of every output, i.e., to ensure that each output is active when the inputs are selected to make it active. However, "inactive" tests of each output (insuring that the output is not active when it should not be) are necessarily limited to those the test designer feels to be high-probability cases. An exhaustive set of inactive tests can be prohibitively large for even a relatively simple function.

The ATG program is intended primarily for the generation of factory tests for the circuit packs, but it proves useful for design verification as well. This program approaches test generation from the same standpoint as the LAMP simulator, as is discussed in detail in a companion paper.[3] One result of this approach to test generation is that the program effectively determines the Boolean function for each output as it actually exists on the pack. By printing these functions in a form so that they can be compared to the functions intended by the pack designer, ATG is an effective tool for design verification of combinational and many sequential packs. By recreating the function from the gate level information, ATG effectively makes both "active" and "inactive" tests of the function design.

2.1.4 Experience

The simulation of the 1A Processor circuit packs for design verification was not a one-time occurrence, but continued throughout the design process. During this simulation procedure, some errors were found on a majority of circuit pack codes. Without circuit pack simulation, these errors would not have been found until complete unit simulation, or perhaps not until testing of the first hardware model of the unit. In most cases, even if the errors were detected at the complete unit level of simulation, the expensive task of generating the artmaster for the pack would already have been completed.

The tests developed for circuit pack design verification served as a substantial portion of the factory tests for these packs. The process of achieving complete fault detection capability uncovers redundancies,

thereby enhancing design verification. Section III describes the fault simulation of circuit packs.

2.2 Complete unit design verification

Inherent in major unit simulation is the need to first verify the accuracy of the simulated unit (or simulation model). This is done by running the diagnostic tests on the simulated unit with no faults, then comparing the simulation results with the expected results for each test. This verification procedure proves useful in three ways: it verifies the functional design of the unit, it verifies the accuracy of the design data base, and it verifies the design and expected results of the diagnostic tests.

2.2.1 Objectives

In the design of large digital units, a time lag exists between the completion of the initial design and the arrival of the first manufactured units. Before the advent of high-speed integrated circuits, the unit was breadboarded, in many cases, to allow continuing design feedback while waiting for the manufactured unit to arrive. However, with the increasing complexity and the higher integration levels of digital systems, it may no longer be feasible to breadboard a digital unit for design verification purposes. LAMP simulation now provides an alternative to breadboarding for units as large as 40,000 gates. LAMP was selected over other alternatives for speed, flexibility, economy, and capability.

The use of simulation has significantly decreased the design interval. Once the initial design is completed, it is easier to make a LAMP model of the unit than to manufacture it. Thus, design verification can begin well before the factory model arrives. As is discussed later, features in LAMP make testing the simulation model comparable to testing the hardware unit itself.

Simulation reduces the number of changes that must be made after the unit is manufactured. Because integrated circuits are being used, changes no longer involve just adding or deleting wires from a wire-wrapped backplane. Now changes may require difficult modifications to printed wire or thin-film circuits or to multilayer backplanes. As the changes become more complex, they also take more time. Critical changes may halt all other debugging until the new change can be designed and implemented. The change facilities in LAMP permit fixes involving a small number of gates and wires (under 50 changes) to be implemented almost immediately. Larger changes can be pre-

pared in less than a day, when required. This means that, with simulation, debugging can continue without significant delay when changes are encountered. By simulating early, many changes can be found and incorporated into the design before the first unit is manufactured.

Diagnostic tests are required for the first and all subsequent units in manufacturing. One objective of simulation is to verify the diagnostic test design before testing the first unit being manufactured. The unit test interval can be reduced significantly if it is known in advance that the tests are correct and that the problem is a malfunction in the unit being tested.

The remainder of this section discusses how simulation is used for logic design and diagnostic test verification, and how it fulfills the objectives of decreased design interval and reduction in change activity after manufacture.

2.2.2 Model building and updating

Figure 1 is a diagram of the simulation process. As the hardware design moves toward completion, the information is encoded into a design data base. This data base is used to generate all information for the LAMP model, the circuit pack and interconnection drawings, the artmasters for the circuit packs, and wiring information for the backplane. When the data base is complete, the simulation model is constructed from the information in the data base. This is done in two stages. First, an LSL-LOCAL[4] description of the unit is generated

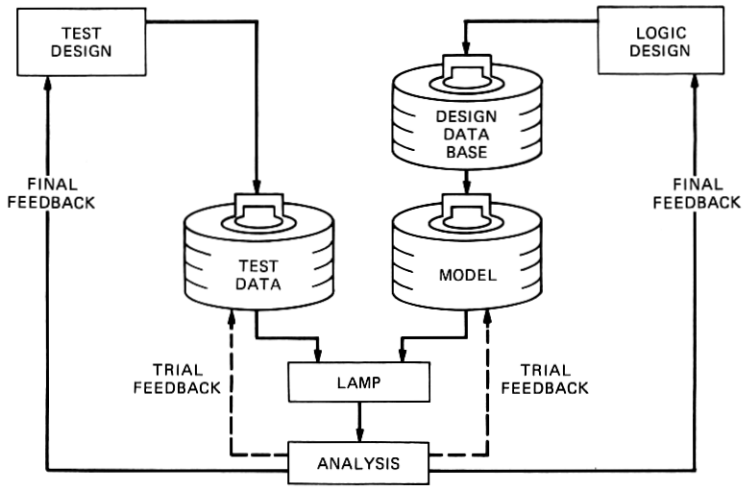

Fig. 1—Design verification.

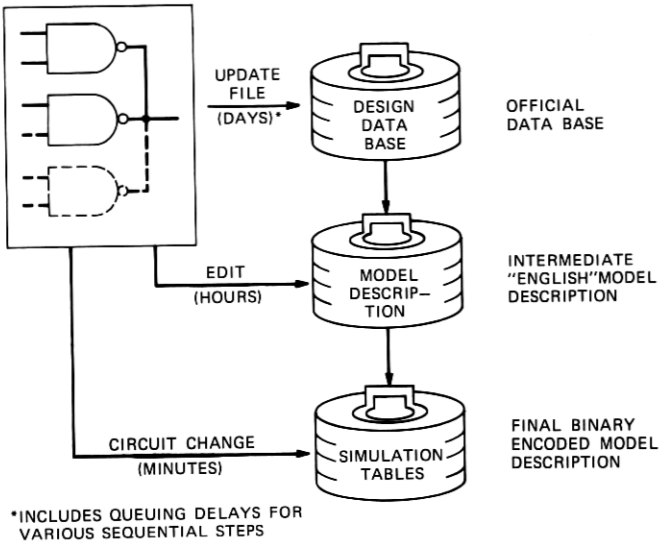

Fig. 2—Model updating process.

from the data base. This LSL-LOCAL description is then compiled into a data set of simulation tables for the model.

The verification of the data base begins as the simulation model is being constructed. Error diagnostics in the programs that create the simulation model find some inconsistencies in the data base. Once the model has been constructed, more errors can be found by running the diagnostic tests.

To make this process practical and easy, methods to change the model must be available. Figure 2 illustrates the change process. To have the change officially issued into the central data base and then to create a new model is a lengthy task, a result primarily of queuing delays in the various sequential steps. The data base has to be updated and a new model must be generated. To shorten the model updating time and to verify changes before they are officially issued, two other methods are used to modify the LAMP model. First, a text editor can be used to update the LSL-LOCAL description of the circuit, which is then compiled into a new set of LAMP tables. Second, the LAMP CKTCHANGE[4] command can be used to modify the existing tables directly. For small units, text editing and recompilation is as fast as using CKTCHANGE. Therefore, this procedure is used for small units, while for large units, because of its speed advantage, CKTCHANGE is used. Any change required is first put into the model by

```
WRITE WØRD (XR), DATA (Ø(52525252))
READ WØRD (XR), EXPECT (Ø(52525252))
WRITE WØRD (YR), DATA (Ø(77777777))
READ WØRD (YR), EXPECT (Ø(77777777))
RUN CYCLE (2)
READ WØRD (XR), EXPECT (0)
```

Fig. 3—Sample diagnostic test.

CKTCHANGE or by editing the LSL-LOCAL description. The change is then verified by further simulation. If corrections are found necessary after simulation, the change is updated and retried. Only when the change is correct will it be added to the official file. This results in fewer changes being made to the official design data base, thus lessening the chance for error.

2.2.3 Simulation procedure

Once a model has been built, tests are needed to verify the correctness of the model. The diagnostic tests are chosen as the main logic verification tests since the test design schedule closely parallels the logic design and, thus, most tests are ready at the time the simulation model is generated. The tests are written in a high-level language from which they are easily compiled into the input language for the simulator. Each test includes an explicit expected result so, while simulating, it is easy to ascertain if the test being simulated is passing or failing. Figure 3 is an example of a simple test. The X and Y registers are initialized and then read to verify the initialization. The run statement executes the test and is followed by a read to determine if the proper action occurred during the test.

Given the LAMP model for the circuit, a set of tests to simulate, and the ability to make quick changes, design verification may begin. The standard procedure is to simulate a complete functional block of tests called a phase. During the simulation, a list of test failures is automatically produced by LAMP. These failures could be hard errors or logic 3 (output in unknown state) propagated to the output. The data from the failing tests are analyzed by the logic designer to determine the source of the errors. Frequently, the solution to a problem is obvious from the test results. In other instances, a follow-up run is required to determine the exact cause of the error.

LAMP provides two basic debugging facilities, an oscilloscope-like timing trace and a stop-and-display feature. Some follow-up runs in-

volve simulating the failing test and generating an oscilloscope-like output trace covering a large number of points. Typically, 100 to 500 points may be traced. This type of trace is effective when the designers know where the error is likely to be. The timing trace is also used extensively with special tests of critical timing functions and complex sequencer interactions. This output has proved to be the most effective technique for uncovering circuit timing problems.

Another type of follow-up run involves simulating the phase in the conversational mode and imbedding stops in the simulator. The stops are activated when a preselected gate changes to a specified value. When the stop is activated, the simulation is suspended, and the logic designer can look at the state of any gate at that instant in time. This allows the designer to trace the trouble back to its source. Imbedded stops are used if the time of occurrence of the problem is known or if the problem is isolated to a particular gating lead changing state for an unknown reason. In the first case, a stop is planted at a particular time. The simulation is run up to this point and then stopped. Using the DISPLAY facility in LAMP, the designer then displays the states of critical gates. The DISPLAY command presents the current value of the gate along with its fan-in or fan-out gates, plus their logical level (value). The values are those at the time the simulation stopped. If these gates are in the wrong state, then the values of the inputs are checked to trace back the problem. The tracing continues until the source of the error is determined. In other cases, the value of a gating or data lead is used to activate the stop. This is used when it appears that the problem is caused by a function changing state at the wrong time. Again, the display facility is used to track the problem back to the source.

Once the trouble is isolated by oscilloscope-like tracing or imbedded stops, either the circuit is changed, if there is an error in the logic design or in the data base, or the diagnostic test is modified, if there is an error in the tests. The simulation is then rerun to verify the correction. These two procedures are so effective that, for some problems, the designers have preferred debugging the logic and tests via simulation even after hardware models are available. Running tests on simulation is slower than running the tests on the unit; however, debugging is facilitated by the ability to suspend the simulation and to observe large numbers of points internal to a circuit pack that are not observable on the hardware model. This has significantly reduced the design verification interval and, by finding the errors during simulation, many costly changes have been eliminated.

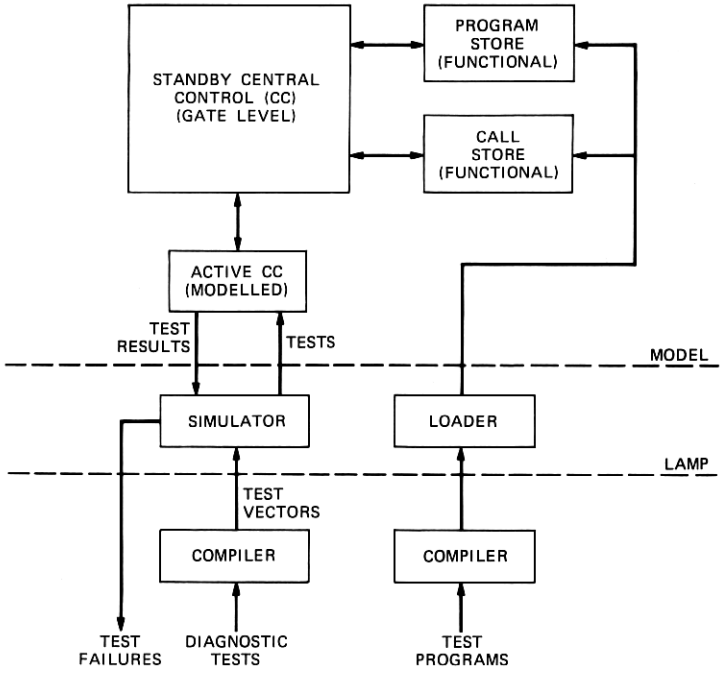

Fig. 4—Central control simulation.

2.2.4 Experience

The central control for the 1A Processor was the first and largest unit whose logic was extensively verified using LAMP. The experience gained in verifying the central control is presented here to illustrate the effectiveness of LAMP for logic and diagnostic test verification. Figure 4 is a functional block diagram of the simulation models and interfaces. The 1A Processor contains duplicate central controls. When diagnostic testing is performed, the active central control tests the standby central control through an interface port. In the simulation process, the LAMP model for the complete standby central control was created. A simplistic model of the active central control was created to take the tests and apply them to the standby central control at the proper time. The active central control model contains an oscillator that could be started and stopped, allowing the circuit to settle and a new test to be applied at the appropriate time. In the active central control model, a comparator circuit was built in to check the actual results of a test with the expected results. Whenever the error flag gate became active, a message was printed listing the failing test. For conversational

simulations, this allowed termination of simulations if the number of failures became too large.

In addition to the diagnostic tests, a series of special test programs was simulated that was designed to test program execution functionally. Since these programs were not written in the high-level language of the diagnostic tests, they were not simulated in the same way. Instead, using LAMP features, a functional model consisting of a program store and a call store was constructed and connected to the gate model of the central control. The programs were compiled and loaded into the memories using a special memory loader facility. Special LAMP vectors were written to initialize the standby central control model and then to release the oscillator. This allowed the clock to run continuously and to simulate actual program fetching and execution. Conversational LAMP control procedures were used to stop the simulation when the program transferred to the error address or when it reached the return address. These programs allowed operational testing in addition to the diagnostic testing.

For the central control verification, approximately 20,000 diagnostic tests and 4000 words of program were simulated. The simulation of all the special tests and most of the diagnostics was completed before the first unit arrived from the factory. Through simulation, approximately 85 percent of the logic and 80 percent of the diagnostic tests were verified. The areas remaining to be tested were primarily the circuitry and tests that interconnect the central control with its system environment. At the present time, this type of testing is beyond the capability of LAMP, mainly because of the large number of gates required to model all the units connecting with the 1A central control.

Many diagnostic and logic changes were generated as a result of simulation. LAMP is an indispensable part of the development process. It has proven effective in reducing development intervals and in reducing the number of circuit modifications.

III. FAULT SIMULATION

3.1 Purposes

Fault simulation is necessary to determine (and enhance) the detection level of the diagnostic tests and can be viewed as an extension of the design verification process. The true behavior of a circuit is studied during verification. The set of other possible behavioral responses can be ascertained with fault simulation. This process is systematic; responses are derived on a fault-by-fault basis. This type of information is essential to the 1A Processor subsystems because (*i*) it contributes

to meeting stringent factory test requirements, and (*ii*) the complete characterization of subsystem behavior is necessary to satisfy long-term in-service system maintenance requirements.

Fault simulation at the circuit pack level is not complicated and is discussed in the next section. On the other hand, unit level simulation is quite complex. Many factors are involved, including large gate counts, precise timing, complex fault modeling considerations, and limited computer resources. These are discussed in the remaining sections.

3.2 Circuit pack simulation

The size of 1A Processor circuit packs (100 to 400 gates) makes fault simulation on LAMP an easily managed process. Simulation models of packs are extracted from the central design data base. These models seldom require any additional modeling changes. Classical faults are simulated for every gate in the circuit. Tests are designed to detect every simulated fault, if possible. A mixture of manual test design and automatic test generation via ATG is used.[3] The process is facilitated through the use of interactive LAMP on the IBM 360/67 or 370/168 computer.

Each test consists of one or more input vectors and an expected set of outputs. The input vectors are simply strings of 1 and 0 combinations that are applied to the inputs of the pack. The tests need not be functionally arranged, but may be independent of each other. Testing is not "clocked," inputs are applied simultaneously, and outputs are examined well after the circuit has settled down. Test minimization is not stressed, since the cost of simulating each fault per test is small, and redundant test sequences may improve detection of nonclassical and/or multiple faults.

As described in Section 2.1, verification tests were generated using a combination of manual and ATG techniques. These tests were then run in the fault simulation mode, achieving an average of 75 percent fault detection. Additional tests were generated with manual and ATG techniques to achieve 100 percent fault detection.

The ATG program was used on about 50 percent of the packs to increase fault detection to about 90 percent although, on some combinational packs (about 10 percent of total), ATG immediately produced 100 percent detection. Manual techniques were required to provide detection of the last 10 percent of the faults on most packs. During this process, design redundancies and other bugs were uncovered. Overall, fault simulation at the circuit pack level produced debugged

factory tests and helped uncover design bugs, thus significantly reducing the intervals required for manufacture.

3.3 Unit modeling

The logic model used for unit level design verification forms the basis for the fault simulation model. In some cases, it may be necessary or desirable to make alterations. Simulated logic used only to support verification studies may be removed to save simulation time. It may be necessary to manually append additional models of specialized circuitry not kept in the design data base. Typical devices are discrete analog components such as operational amplifiers, current drivers, and signal buffers. This circuitry communicates with the logic and affects its state, but is nonlogic in nature. Often, many one-of-a-kind models must be constructed through truth table and timing diagram studies. In some cases, circuitry exists that is not modeled. The 1A Processor call/program store core memory is an example. While functional simulation can be used to model the memory for design verification at the present time, LAMP cannot support fault list propagation into and out of a functional model.

Once the fault model is completed, circuit initialization must be considered prior to simulation. Ideally, simulation should begin with the circuit in an unknown state. This represents the most accurate approximation to the physical circuit whose state prior to testing is not necessarily predictable. The unknown state approach is normally used for design verification simulation, but not for fault simulation. This is because starting from a known state significantly reduces the excessive simulation CPU time caused by the potential for large fault list buildup during initialization. The resulting loss in accuracy is small.

A known state is normally achieved by applying an initializing sequence to the circuit using true value simulation. Fault simulation is conducted starting with this true-value state and a set of "null" fault lists.

3.4 Test selection

Test selection is an important consideration for large unit fault simulation because it is costly to simulate an input vector, and each test usually expands into a series of from two to ten such vectors.

The decision concerning which tests to simulate is influenced by the particular objective, the circuit model, and the available computer resources. Fault simulation to evaluate early tests or to support the

improvement of tests (test enhancement) is concerned with the detection of classical faults. This objective permits the exclusion of tests designed to improve fault isolation or to detect lead shorts (such as a walk of 1 through a field of 0's). Test exclusion is also necessary to size the tests according to the capabilities of the simulation model. This is most evident in tests that deal with the system environment, such as tests of "interunit" communication buses or of nondigital circuitry such as memory. A set of tests may also be simulated versus only a fraction of the faults. This is discussed further in Section 3.6, under "simulation strategy."

How the tests are sequenced is also important. Functional ordering (grouping tests according to the circuitry being tested) is essential to make the tests useful as a repair vehicle. The test phase is considered the basic functional entity whose predesigned sequence must be maintained. In some cases, tests may be deleted from a phase during simulation, but the order of remaining tests is preserved. During test evaluation, phase ordering is not essential. How or when a fault is detected is of little consequence at this stage. Functional phase ordering assumes greater importance when data are collected to support trouble location dictionaries. Also, tests excluded from earlier stages of simulation are included here wherever possible.

3.5 Fault selection

The 1A Processor uses conventional TTL integrated circuits for logic function implementation. Discrete circuitry augments this logic where required. The major subsystems use LAMP to simulate "stuck-at" (input open, output stuck at 0 and at 1) logical faults on TTL gates. Shorted fault simulation has, to date, not been used at the unit level. The restriction to stuck-at fault simulation is an engineering decision. In an actual circuit, the possible failure modes are much more extensive, including not only lead shorts but also timing changes, voltage changes, intermittents, etc. It is assumed that the majority of these faults will behave as stuck-at faults for a portion of the tests. Experience with previous electronic switching systems has shown this assumption to have validity.

Since there are several hundred thousand possible stuck-at faults in 1A Processor subsystems, the fault selection process must be facilitated by various options available in LAMP. Depending upon application, faults are selected on an individual basis by gate name, by circuit pack, or to a certain extent by hardware function.

When detection information is being sought, random sampling is used. A sufficiently large random sample has been found to predict detection levels accurately. According to requirements, samples are selected on a uniform, localized, or stratified basis.

Several automatic fault administration options in LAMP have reduced 1A Processor simulation costs significantly. Early termination is used extensively during test evaluation and enhancement. Under this option, a fault is terminated (removed from simulation) after one "hit," or detection. Typically, this option saves from 50 to 75 percent of simulation time. Fault collapsing removes $n - 1$ out of n logically equivalent faults from the fault set being simulated. For example, if five faults cause the same effect upon a logic circuit, LAMP simulates one of the five. Typically, this option reduces the fault set by 25 to 50 percent. Undetectable fault elimination, which removes faults such as those on unused gates, reduces fault sets up to 10 percent.

Even after every attempt to prune the fault set has been exhausted, it is still necessary to form fault partitions for typical large unit simulation runs. The degree of partitioning necessary depends upon the size of the unit, the tests, the circuit topology, and the computer resources. The primary factor influencing fault partitioning is the available computer main memory. Using an IBM 360/67, typical fault sets are in the 3000- to 6000-fault range. Using an IBM 370/168, sets as large as 30,000 faults have been simulated.

On the other hand, test partitioning (the smallest partition being a phase) is primarily influenced by simulation time limits. Some partitions may require 1 or 2 hours of processor time.

For protection, the LAMP CHKPOINT/RESTART facility is used to permit rollback in the event of a computer, simulator, or procedural failure.[4]

3.6 Simulation strategy

An interactive simulation procedure is used in the 1A Processor for test enhancement. A random sample of faults is selected and simulated with the diagnostic tests. The results are analyzed to reveal undetected faults for which additional tests are designed. A new sample is then selected, consisting of the previous undetected faults plus an additional random sample, and the process is repeated. The size of a fault sample is generally larger than the minimum required to predict detection levels. This increases the probability of revealing classes of undetected faults. Ultimately, large classes are eliminated, and a point of diminish-

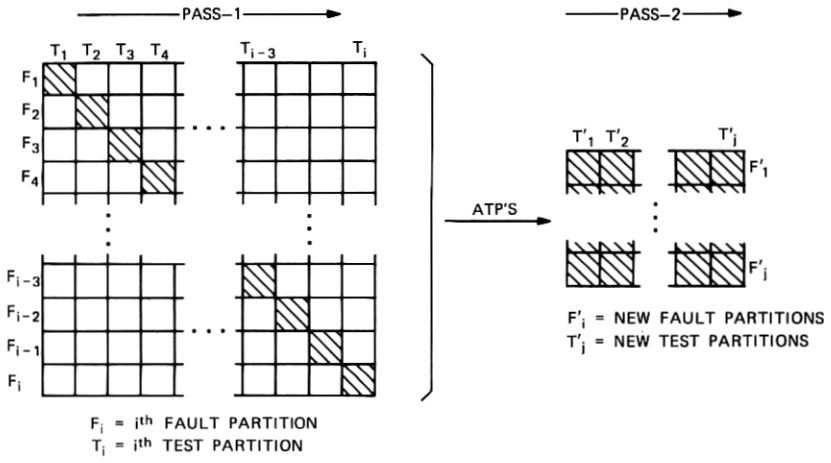

Fig. 5—Two-pass diagonal simulation.

ing returns is reached. At this stage, complete fault detection information is secured by simulating all remaining faults with the complete set of tests.

Selection of several fault partitions (such as a random sample divided into four parts) that must be simulated against many phases, a phase at a time, requires the execution of many successive simulation runs. The most obvious way to execute these runs is to simulate the various combinations of fault and test partitions in succession. This approach does not prove efficient, and significant cost reductions are possible through an alternate strategy called "two-pass diagonal simulation." Figure 5 is an illustration of this method. In the ideal situation, i mutually exclusive fault partitions and i test partitions are selected. Each fault partition is chosen in the area of circuitry that a corresponding test partition is designed to exercise. This correspondence significantly increases the probability of fault detection per second of processing time. Pass 1 simulation then consists of i runs, not i^2. Early termination is used to remove detected faults prior to pass 2. In pass 2, undetected faults are collected and repartitioned into a smaller set of j runs, and all combinations of partitions are simulated. The effectiveness of this method lies in the fact that, for a minimum of resources, the great majority of faults are detected in pass 1. From a practical point of view, mutually exclusive fault sets with clear test associations are hard to produce. Instead, overlapping fault partitions are usually selected. Pictorially, this means that, in Fig. 5, regions off the major diagonal would be lightly shaded. This reduces slightly the

efficiency of the method, but the savings are still significant. Results have shown that tightly connected circuits benefit least from this technique because the concept of localized fault detection tends to break down.

The method just described is used primarily for simulation where detection information is being sought. In other applications such as data collection for trouble location dictionaries, the procedure of Fig. 5 is not used since it produces incomplete fault behavior information. In such cases, the simple strategy of simulating all fault and test partitions may be utilized with corresponding CPU time increases.

3.7 Experience

A composite LAMP model of the 1A Processor call/program store was constructed to support design evaluation and diagnostic development. Figure 6 is a block diagram showing interrelations of distinct portions of this model.

The model contains approximately 10,000 LAMP gates. This count is about 40 percent higher than the real gate count because of attendant logic controlling bus interaction and specially modeled "nonlogic" circuitry. This model was verified in the manner described in Section II. A preliminary diagnostic was designed consisting mainly of functional exercise tests, which translated into about 3000 LAMP input

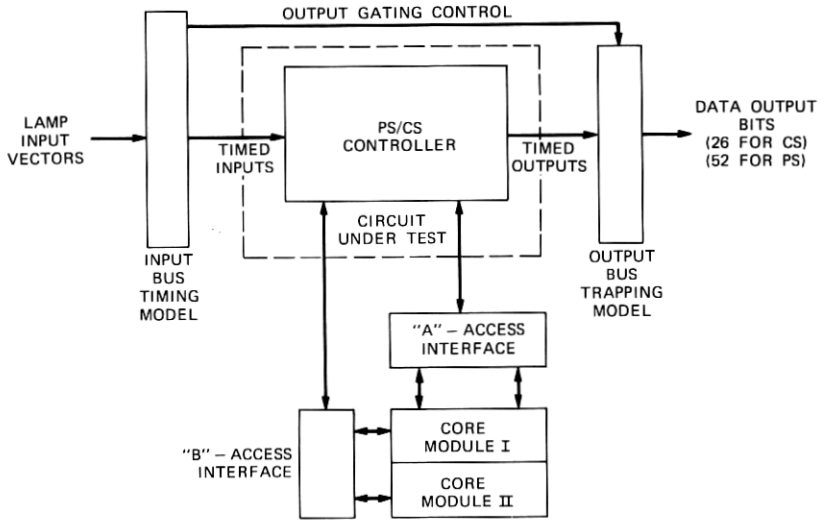

Fig. 6—Program/call store lamp model.

Table I — Program store simulation data

Statistic	15,000 Faults	580-Fault Sample
Number Detected*	10,400	400
Number Race Faults	2200	83
Number Oscillation Faults	80	3
Detection with Races Removed	81%	80%
Detection Estimate Following First Test Enhancement Iteration	90%	90%
Estimate of Ultimate Detection Level	96%	96%

* Using initial set of tests.

vectors distributed over a number of phases. At the time, the inability to model the core memories with LAMP gates necessitated omission of several test phases.

A study was conducted using the model of Fig. 6 and these tests to meet the following objectives:

(*i*) Determine the detection power of the tests.
(*ii*) Provide data to support test enhancement.
(*iii*) Evaluate proposed methods of data collection for use with other units (this was a pilot study for large unit fault simulation).

The model contained a set of 15,000 meaningful classical faults. As a first experiment, this set was simulated using the two-pass diagonal simulation method of the previous section. As a second experiment, a random sample of 580 faults was selected and simulated against all tests. In the first experiment, the number of faults in each partition was chosen to optimize use of the host computer. For the random sample, it was possible to simulate the 580 faults in one partition. About eight partitions were selected for the 15,000-fault experiment, each containing on the average 4000 faults. Significant overlapping of fault partitions was necessary because the minimum faultable unit was selected as a circuit pack (out of convenience) which often encompassed several functions.

About 24 CPU hours of IBM 360/67 computer time were required to complete the above experiments. This included the time required to restart several runs because of procedural errors and system crashes. Through some additional runs, it was possible to show that diagonal simulation saved about 50 percent of the CPU time that would have been required to simulate all tests versus all faults.

Table I contains some simulation results and estimates for the two

experiments. It is important to emphasize the fact that the tests were an initial cut used to support early factory testing of frames. The resulting detection level of 81 percent was about as expected at this stage of design. The most significant result was the relatively large number of race faults encountered. The LOGIC simulator[5] was used for the study to save computer time. This resulted in race faults (those causing indeterminate gate or output states) and oscillation faults (those leading to circuit oscillation) being removed during simulation when encountered. Subsequent studies using the FAULT simulator[5] showed that most race faults were in reality noncritical races that did not cause indeterminate output states. Furthermore, the race faults were actually detected at about the same level as normal faults. In Table I, it was, therefore, reasonable to remove them from the sample in computing the actual detection level.

Oscillation faults are a serious problem that demand careful study. They potentially jeopardize system operation by causing interference with other units on the system buses. Although such faults would probably be detected in the real system, they were not considered as such in Table I.

The undetected faults in the random sample were carefully analyzed and led to some interesting results. Fourteen major classes of undetected faults (with respect to the 15,000 fault sets) were categorized. In most cases, these classes consist of similar faults on repetitive functions, such as successive bits of a register, which require a few tests for detection. In some cases, other faults were revealed that implied the design of a class of new tests. Several one-of-a-kind faults, each requiring unique tests, were also revealed by the analysis. About 2 percent of the faults were undetectable for various reasons. Sixty percent of the faults were associated with operational logic, and the rest with maintenance circuitry.

It is estimated in Table I that new tests designed because of the results of the random sample analysis will reduce the undetected fault set to about 10 percent. It is also estimated that, through the use of LAMP in this iterative process, no more than three iterations will be required to achieve an ultimate detection level of 96 percent of the classical faults.

Resolution of the remaining 4 percent, which are truly undetectable, pose a problem. Some represent true circuit redundancies (in the simplest case, a single gate output feeding a gate twice). Others, more subtle, reside in circuitry used to improve noise or electrical margins.

These might be detected under worst-case conditions by existing tests. A third class deals with system constraints (inputs constrained not to assume certain state combinations).

It is impractical to consider complete removal of these faults through design changes. In any event, LAMP has done its job by categorizing these faults. Maintenance information can at least be provided to help deal with the possibility of their existence throughout the life of the system.

IV. CONCLUSION

When LAMP was first introduced, it received almost immediate acceptance and support from circuit and diagnostic program design groups, although many growing pains were involved with its use. On countless occasions, the user community taxed both LAMP and its host computer resources to their limits. In response to this, LAMP has grown and matured, making significant improvements in capacity, speed, and capability.

The use of LAMP to verify the paper design of 1A Processor subsystems significantly reduced laboratory debugging intervals and provided major cost reductions. Logic design errors were located and corrected prior to the construction of initial hardware. In association with this, the "first iteration" of diagnostic design, the debugging of functional tests using LAMP simulation, was completed prior to the availability of system laboratories. These tests were then used to test the frames in the factory environment.

The availability of interactive LAMP has been a significant aspect of design verification. The option to freeze the state of a logic simulation in order to examine internal nodes has proved so powerful that circuit designers have sometimes preferred this facility to the actual unit as a debugging tool.

The LAMP fault simulator has been essential to the development of complete circuit pack test vectors for the 1A Processor. The extension of fault simulation to large subsystems is just beginning. Initial fault simulation studies using the 1A Processor program/call store are encouraging. Iterative test enhancement using LAMP will insure the detection or classification (as to reason for not being detected) of every stuck-at logical fault. This is of primary importance because of the very stringent maintenance requirements of the 1A Processor.

In the future, the trend toward higher scales of logic integration will increase the use of LAMP for design verification and factory test

development. The use of LAMP as a breadboard will become a practical necessity.

REFERENCES

1. R. E. Staehler, "1A Processor—A High-Speed Processor for Switching Applications," International Switching Symposium Record, June 1972.
2. H. E. Vaughan, "An Introduction to No. 4 ESS," International Switching Symposium Record, June 1972.
3. S. G. Chappell, "LAMP: Automatic Test Generation for Asynchronous Digital Circuits," B.S.T.J., this issue, pp. 1477–1503.
4. H. Y. Chang, G. W. Smith, Jr., and R. B. Walford, "LAMP: System Description," B.S.T.J., this issue, pp. 1431–1449.
5. S. G. Chappel, C. H. Elmendorf, and L. D. Schmidt, "LAMP: Logic-Circuit Simulators," B.S.T.J., this issue, pp. 1451–1476.