NASA Technical Memorandum 4241

Analog-Digital Simulation of Transient-Induced Logic Errors and Upset Susceptibility of an Advanced Control System

Victor A. Carreno, G. Choi,
and R. K. Iyer

NOVEMBER 1990

NASA

NASA Technical Memorandum 4241

Analog-Digital Simulation of Transient-Induced Logic Errors and Upset Susceptibility of an Advanced Control System

Victor A. Carreno
Langley Research Center
Hampton, Virginia

G. Choi and R. K. Iyer
University of Illinois
Urbanna, Illinois

National Aeronautics and Space Administration
Office of Management
Scientific and Technical Information Division

1990

Abstract

This paper describes a simulation to predict the susceptibility of an advanced avionics control system to electrical transients resulting in logic errors, latched errors, error propagation, and digital upset. The system is based on a custom-designed microprocessor, and it incorporates fault-tolerant techniques. The system being tested and the method of performing the transient-injection experiment are described. Results for 2100 transient injections are analyzed and classified according to charge level, type of error, and location of injection.

Introduction

Digital system upset, as used herein, is caused by electrical transients and refers to the mode that a system can enter in which the system is not performing its intended function but in which no physical component failure is caused by the transient. This upset mode is inherent to digital computers and other systems with discrete internal states. It is estimated (ref. 1) that 80 percent of computer "crashes" are caused by electrical transients. Digital control systems onboard aircraft and spacecraft are exposed to many sources of electrical transients. Of particular interest are electrical transients produced by sources external to the system like lightning, high energy radio frequency (HERF) transmitters, electromagnetic interference (EMI), and high energy particles.

The application of digital systems to the control of critical functions necessitates the study of system susceptibility to upset by electrical transients. Upset studies have been performed in the past using hardware injection experiments (ref. 2) and simulation injection experiments (ref. 3). These hardware and simulation experiments were performed on identical systems, and the results were compared. The system test-bed was a general-purpose computer based on an Intel 8080 microprocessor. This system, however, was not representative of avionics control systems (electronic systems used in the control of aircraft functions), and the program that was executing during injections was not a real application program.

The study of transient injections on digital systems can yield two types of results—the understanding of the mechanisms by which electrical transients produce logic errors, error propagation, latched errors, and possible prevention strategies; and the impact of these errors on the overall functionality of the system and the possibility of system upset. Given that there is a logic error in the system, protection schemes can prevent the system from entering an upset mode. These protection schemes can be implemented by using specially designed hardware and software algorithms. This paper describes a simulation study of transient-caused logic errors, latched errors, error propagation, and pin errors (errors propagating to external pins) on a microprocessor designed specially for avionics applications. The section "Simulation Methodology" describes the selected simulation program and the development of the program into the transient-injection and error-propagation analysis system. The section "Experiment Setup" describes initial runs to determine simulation accuracy, injection waveform, gate-transistor combination requirements, and the processor used as the system under test. The simulated processor, the HS1602, was developed by Hamilton Standard and is the computational unit of an electronic engine controller (EEC). Detailed descriptions of the processor and two hardware units of the EEC were obtained from Hamilton Standard. Results for 2100 transient injections are presented in the section "Experimental Results." An error-transition model is described in the section "Data Analysis," and a description of the feasibility of performing such assessment by using digital simulation is presented in the section "Upset-Assessment Outlook."

Simulation Methodology

The modeling of the transient-injection process (fig. 1) requires multiple levels of simulation. Circuit analysis is employed to obtain the incidence of logic errors caused by the analog transients. Logic errors are then used in a logic level simulation (gate, functional) to determine error-propagation characteristics of the entire system under test. The SPLICE1 (simulation program with large-scale integrated circuit emphasis) mixed-mode simulator selected for the study can perform coupled-circuit analysis and logic simulation. A mixed-mode simulator eliminates the need for two separate programs and the interfacing associated with data transfer, interpretation, and manipulation between the two programs. The SPLICE1 consists of the following three parts:

1. The language (description) translator, which is similar to a compiler. The description listing of the circuit is analyzed for syntax errors, undefined arguments, connectivity, etc. A data file is generated that represents the circuit with initialization and run-time parameters.
2. The simulator, which takes the circuit data file and performs the actual simulation.
3. The postprocessor, which plots and prints selected node values against time from the simulation.

The detailed implementation and characteristics of the SPLICE1 simulator are documented in reference 4.

Several modifications and enhancements have been made to the SPLICE1 simulator for the transient-injection study. Typical injection of transients is performed by connecting a current or voltage source at the node where the injection is to occur. This method, however, requires the reprocessing of the entire circuit for changes in injection location and current characteristics. A new method was developed to allow the direct specification to the simulator of injection locations and electrical characteristics of the transient. This was accomplished by changes in the SPLICE1 relaxation algorithm (ref. 5). The electrical characteristics of the transient are then specified as a mathematical expression as a function of time.

The SPLICE1 simulator was also enhanced by the creation of a utility to manipulate external data files during simulation to service modules that represent external memory. Modeling large memories (random-access memory (RAM) and read-only memory (ROM)) with SPLICE1 primitives (models predefined in the SPLICE1 description) was not feasible. This utility allows the user to specify a memory device as a normal electrical device with all connections and electrical characteristics associated with it. When the simulator processes the memory module during a memory access, the address is calculated and an external file is used to store or retrieve the value for the calculated memory location.

Finally, two new facilities were developed for the output postprocessing. The first facility extends the output of SPLICE1 from 16 to 80 nodes and produces an output only if any of the 80 selected output nodes changes value. It also converts the value of selected groups of nodes to a hexadecimal radix. This notation can aid in the identification of microprocessor operations by grouping the DATA and/or ADDRESS lines. The second new facility is an extended-trace facility that monitors all the nodes in the system, approximately 4000 nodes for the HS1602. Output is generated for a node every time a gate driving that node is evaluated by the simulator. Monitoring all nodes is essential for a study of fault propagation.

Experiment Setup

Processor Description

The EEC's are typically full-authority, dual-channel jet-engine controllers. They incorporate fault-tolerant techniques at various levels. One channel of the EEC, excluding the interfaces, was simulated and used as the system being tested. An EEC application program was supplied by Hamilton Standard and was modified to account for the single-channel nature of the system being tested. The simulated EEC is based on the HS1602 16-bit microprocessor. Integral to the microprocessor are a parity generator and parity-error detection, a watchdog timer, and a universal transmitter and receiver. Other modules are the arithmetic logic unit (ALU), control unit, decoder unit, countdown unit, and multiplexer. A gate-level description of the entire microprocessor was supplied by Hamilton Standard. The gate-level description is structured in a hierarchical manner. Each basic submodule of the gate-level description was then described at the transistor level. The simulation can be performed with any combination of gate-level submodules or transistor-level submodules. An increase in transistor submodules results in a penalty in simulation execution time. Transistor models for the transistor-level submodules are CMOS (complementary metal-oxide semiconductor) models similar to those found in analog circuit analysis tools, such as SPICE (simulation program with integrated circuit emphasis). Silicon parameters for the transistor models were obtained from the manufacturer. Capacitance loads were calculated from metallization lengths in the circuit layout. Metallization lengths supplied by the manufacturer were from VLSI (very large scale integration) CAD (computer-aided design) tools.

Injection Waveform

The current waveform used for the injections is the double exponential type, based on the following approximate analytical solution (ref. 6) for ion track charge collection:

$$i(t) = A\left[e^{-t/\alpha} - e^{-t/\beta}\right]$$

where α is the collection-time constant and β is the track-establishment constant. The values of α and β were set to 1.64×10^{-10} and 5.00×10^{-11}, respectively, for the injection experiment. The constant A is the approximate maximum current, and it was varied to produce charge accumulations of 0.5 to 9 pC. The charge level was limited to 9 pC, since charge levels higher than 9 pC cause permanent damage to the circuits (ref. 7). It was found, however, that for the CMOS technology modeled in this experiment, it was total charge, and not the time constants α and β, that determined the potential of the transient to cause logic errors.

During transient injections, the simulated EEC was executing the initialization portion of the application software. The initialization consists of the watchdog test, parity test, instruction set test, RAM test, and ROM sum test. Also, initial data values are loaded in RAM during this part of the program.

Initial Runs

To establish simulation accuracy and increase confidence of results, an undisturbed simulation run was compared with the operation of the actual hardware unit. Two comparisons were made. The first verifies the correct flow and execution of instructions by monitoring data, address, and control lines during the time that these lines are stable (valid). Correct execution flow was observed for 74 instruction cycles (90 300 simulation steps). The second comparison, which is more rigorous, monitors all changes in logical values, including transitional times. For the second comparison, 16 231 simulation steps were analyzed. This comparison reveals electrical characteristics of the simulation, such as propagation delay, race conditions, and gate loading. This comparison, however, is not completely deterministic, since some performance parameters in the actual hardware circuit, given the same conditions, vary from one experiment to the next. Gate delay, for example, can be measured and specified as minimum, typical, and maximum delay time. Differences in transitional times between the simulation and the actual system were 3 nsec or less after normalization. Normalization was needed because the actual system runs at 12.08 MHz and the simulation was set at 82 nsec per clock cycle (12.195122 MHz). Normalization consists of multiplying the simulation times by a 1.009530 factor. Simulation resolution was 1 nsec; therefore, all time values were in 1-nsec increments. Differences in transitional times are well within the acceptable levels of accuracy for the simulation.

Gate-Transistor Module Combination

For transient injections, the number of transistor-level submodules needed for accurate simulation was determined by experimentation. Simulations were performed with all gates at 1, 2, 3, 4, and 5 gate distances modeled at the transistor level, where gate distance is as defined in figure 2. For a combinatorial circuit, it was found that a minimum of three gate distances is needed for accurate simulation; that is, modeling gates at the transistor level at more than three gate distances does not change the results of the simulation.

When latches in the circuit are within three gate distances of the transient-injection node, a minimum of four gate distances is required for consistent results. Figure 3 shows an example of a circuit with a latch at two gate distances from the transient. In this example, the shaded gates and latch need to be modeled at the transistor level for correct results. The transient injections for the gate-distance experiment were at the maximum 9-pC level.

Experimental Results

A total of 2100 transient injections were performed on the simulated processor. Seven nodes were randomly selected from each of the six major units. The nodes were injected with 10 electrical transients, with charge levels of 0.5, 1.0, 2.0, $\cdots$, 9.0 pC, at five different times during program execution. The incidence of errors, as a function of charge level, is shown in table 1. A first-order error is defined as a logic error that is present one clock cycle after the transient has been injected (i.e., latched error). Second-order errors are latched errors that are present two or more clock cycles after the transient. First-order pin errors are first-order logic errors that appear at the device pins. First-order pin errors are a subset of first-order latched errors. Functional errors were recorded when there was a change in the functional operation of the microprocessor.

Injections with charge levels of less than 2 pC cause few or no errors. For charge levels from 2 to 6 pC, there is an increase in all types of errors with increase in charge. Increasing the charge level above 6 pC does not have any significant effect on the number of errors.

Table 2 shows first-order errors in the system for injections in each individual functional unit. First-order error is an approximation of the susceptibility of the module to the transient injection. As can be seen from the table, the watchdog and the ALU have the highest susceptibility to transients, and the decoder, control, and countdown have about half the errors. Injections in the multiplexer did not produce any effect on the system.

Because of the complexity of the software used in the EEC and the various recovery mechanisms implemented in software and hardware, a quantitative measurement of system upset was not feasible with digital simulation. During simulation, however, many of the mechanisms for recovery were observed. "Power on" resets were issued after some errors were detected by the processor as parity error. This reset causes reinitialization and check of the RAM, PROM (programmable read-only memory), etc. When the processor detects a parity error, a parity-error counter is incremented and the suspected memory unit is "charged" with the parity error. The parity-error counter could be used at a later time to change control of the engine from one channel to another. Another observed functional recovery was corrupted data values that were discarded because the values were out of range.

To thoroughly evaluate the recovery mechanism and determine if the system would enter an upset mode after a functional error, the simulation would have to be run for several million machine cycles, and the functional outputs of the system would have to be monitored to determine if they were within a specified range. This translates into several weeks of simulation time for each transient injection, which makes the creation of a data base not viable. A simulation run of 10 msec real time, with 12 two-input logic gates and 2 D-type flip-flops modeled at the transistor level, requires approximately 130 hr of CPU time in a MicroVAX II computer.

Data Analysis

This section presents an analysis of the error-propagation probability by using an error transition model. That is, given an error in module A, what is the probability that the error source was module B? The results of this analysis are useful in identifying several critical aspects of the system: the identification of the critical-error propagation paths; the determination of the module most sensitive to error propagation; and the module with the highest potential for causing external pin errors. Figure 4 is a state transition diagram, based on the measured data, to quantify the intermodule latched-error propagation. A state in the figure represents a system module that contains one or more errors. In this model, the system can be in more than one state at the same time. Given that errors exist in a module, the numbers inside the state are average numbers of errors. When first-order latched errors exist in a module, they are not necessarily the results of an injection in that module. For example, although injections in the multiplexer did not cause any latched errors, first-order errors were observed in the multiplexer after transient injections in other modules. This is shown by the line from "FAULT INJECTION" to "MULTIPLEX" (0.04) in the figure.

Latched errors in the decoder unit do not propagate from other functional units. All latched errors that occurred in this unit were a result of the direct effect of the injected transient. The probability of a latched error in the decoder unit propagating to the pins is small (0.01). Thus, the decoder is not a critical unit from a fault-propagation point of view. The model shows that the critical fault path in the system is between the control unit and the watchdog unit. Given a latched error in the control unit, the probability that it propagated via the watchdog unit is 0.32. Conversely, the probability of the control unit being the source of a latched error in the watchdog unit is also high (0.30). Although the one-way propagation probability is high in some cases (e.g., 0.63 from the watchdog unit to the multiplexer), none has a higher two-way propagation probability. Therefore, if all other factors are equal, the best way to reduce intermodule error propagation is to protect the interconnections between the watchdog and control units. Since a significant number of functional errors result from the second-order and higher order latched errors, the system level impact of providing this protection is expected to be a decrease in the probability of functional errors.

The model also shows that the modules with the highest potential to cause external pin errors are the watchdog and control units. Fifty-seven percent of all pin errors were a result of the latched errors in the watchdog and control units. The module most sensitive to fault propagation was the ALU. Of all functional units, an error occurrence in the ALU is likely to lead to the largest number of latched errors (9.89). Applying internal retry to ALU operations may be a successful way of reducing the number of latched errors. It is important that, with the exception of the decoder, the probability of the transient injection directly causing the latched errors is low. More than 95 percent of latched errors were caused by propagations from other latched errors in the ALU, control, countdown, watchdog, and multiplexer. Also, 89 percent of pin errors were caused by latched-error propagation; therefore, error propagation is critical in the study of highly reliable systems.

Upset-Assessment Outlook

Testing the resiliency to upset of a hardware unit independently from its operating system software is not meaningful because of the interdependency between the hardware modules and the software program controlling the hardware resources. The application program is also usually involved in the hardware management and forms an integral part of the system and the upset-assessment process. (Application program is used in this context as the program that implements the control laws in a control system.) For example, electrical transients can be detected by the microprocessor as sensor failures. The microprocessor can synthesize the missing values from other sensors or transfer control to other channels and shut itself down or enter other modes. Figures 5 and 6 show hypothetical flow diagrams to illustrate this example. Figure 5 shows no fault-tolerance techniques, while figure 6 shows some built-in fault tolerance. In figure 6, the validity of the reading from sensor A is determined based on other physical values as illustrated by the comparison $g(x1, x2, x3) <$ value sensor A $< f(x1, x2, x3)$, where $g()$ and $f()$

are functions of other sensors or calculated values. In this example, if the value sensor A is out of range two consecutive times, the sensor is assumed to be failed and other paths are taken to calculate control parameters. Physical characteristics of the control loop and control laws are therefore used to manage hardware resources.

Previous work in the area of upset (ref. 8) identified an upset dependency on software. This relation, however, was at a low level of software (i.e., processor instruction level). In this previous work, "illegal" loops were found within the correct program. The illegal loops are a result of multiword instructions. Multiword instructions typically have one, two, or three words. The first word is the instruction itself, and the second and third words are the data necessary to perform the instruction. The second and third words of a multilevel instruction are never read as an instruction under normal operation. If the processor is forced (e.g., with an electrical transient) to read a data word as an instruction, the processor can enter an illegal path or loop, where it no longer is executing the code correctly. When the system is in this mode, reset or other corrective action is usually necessary. The HS1602 instructions, however, are single-word instructions, and no illegal loops exist. The upset-software dependency is at a higher level, and the application and operating system are the determining factors in the upset susceptibility of the system, as illustrated by the example in this section.

During transient-injection runs, the system did not enter an upset mode. The state of the system, however, was in many cases different than the state of the system for the undisturbed (gold) run. Within the scope of this experiment, it is impossible to predict in a quantitative manner the impact that present system state would have had on the continued operation of the system. Limited simulation runs and informal predictions based on observations of the operation of the EEC system suggest that the occurrence of upset is an unlikely event. Thus, random transient injection is not an effective method of uncovering upset in such systems. This situation is similar to software testing for the detection of code design errors. In software testing, like in upset testing, exhaustive testing is not an alternative because of the large state space, and partial testing only reveals the more probable errors. Also, upset testing, unlike normally executed software, has to account for entry into sections of a program at any random location. Normally executed software has predetermined entry and exit points, as defined by test (comparison) instructions. The possibility of random entry into the code makes program flow analysis an extremely difficult task, and the result of the program flow analysis is limited by some of the assumptions that were made to make the problem tractable. Other mechanisms that contribute to the complexity of program flow analysis include watchdog timer updates; hardware-implemented, interrupt-driven, program flow control; hardware status checks; and corrective action during background program execution. Upset, a functional characteristic, can be separated between hardware and software in only the most simple of systems, when the results are almost trivial. Therefore, the study of upset on systems that incorporate recovery mechanisms, multiprocessing, and fault-tolerant techniques requires the analysis of the dependency of the code on the computational and combinatorial units and on the physical mechanisms that cause changes in logic values and disruption of the data flow. Upset assessment of these systems could be performed by analytical means (i.e., program flow analysis, timing analysis, etc.) or a combination of analysis and digital simulation, where the analysis is used to extrapolate the results of the simulation. In systems with recovery mechanisms, deviation from an undisturbed path (compared with a gold system) does not imply that the system is not meeting its requirements. A prediction of the future state of the system can be attempted, by analytical means, from the present state of the system, which was obtained from the simulation after transient injections.

Overall system functional testing also presents the problem of input-output dynamics. Since the control system is an element in a control loop, any control command issued by the control unit is fed back to its inputs in the form of parameter measurements and is used for following computations. Static inputs during upset testing will likely result in inaccurate results; thus, evaluation of the system should be performed in a closed loop. It is rarely possible or economically feasible to test a control unit in its normal control loop. Computer simulation of the mechanical devices in the loop is an alternative for closed-loop testing.

Concluding Remarks

A system for the analysis of electrical transient injection and error propagation was developed based on the SPLICE1 (simulation program with large-scale integrated circuit emphasis) multimode simulator. An electrical engine controller was used as the candidate system for the tool development and injection experiments. The gate-transistor level mix for accurate results was obtained experimentally. For the modeled technology, no more than four gate distances need modeling at the transistor level. Approximately 25 percent of the electrical transient injections resulted in logic errors. Some modules were more susceptible than others to transient-induced logic errors, ranging from approximately 37-percent error incidence for the ALU and watchdog to 0-percent incidence for the multiplexer. These results suggest that error recovery techniques can be applied to the more vulnerable modules in a discriminative manner to obtain a larger improvement from the added cost associated with implementing the error recovery techniques. Error recovery techniques that are implemented at a low level can be used to keep logic errors from propagating to other modules.

In general-purpose systems that do not employ fault tolerance, the study of upset is performed by analyzing the internal states of the system. Internal state monitoring through simulation studies does not lead to upset susceptibility predictions on systems that incorporate recovery mechanisms and fault-tolerance techniques. Fault-tolerance techniques have indirectly addressed some of the weaknesses that permitted system functional upset. Theoretical development is needed in the area of hardware, software, implementation, and control-law dependency to further investigate the upset phenomenon. A "unified" theory will contribute to the study of upset and other areas of great interest, such as ultra-reliable systems.

NASA Langley Research Center
Hampton, VA 23665-5225
September 14, 1990

References

1. Iyer, Ravishankar K.; and Rossetti, David J.: A Measurement-Based Model for Workload Dependence of CPU Errors. *IEEE Trans. Comput.*, vol. C-35, no. 6, June 1986, pp. 511–519.
2. Belcastro, Celeste M.: *Digital System Upset—The Effects of Lightning-Induced Transient on a General-Purpose Microprocessor.* NASA TM-84652, 1983.
3. Carreno, Victor A.: *Upset Susceptibility Study Employing Circuit Analysis and Digital Simulation.* NASA TM-85822, 1984.
4. Saleh, Resve Aslam: *Iterated Timing Analysis and SPLICE1.* M.S. Thesis, Univ. of California, Berkeley, 1983.
5. Duba, Patrick: *Transient Fault Behavior in a Microprocessor: A Case Study.* M.S. Thesis, Univ. of Illinois at Urbana-Champaign, 1989.
6. Messenger, G. C.: Collection of Charge on Junction Nodes From Ion Tracks. *IEEE Trans. Nucl. Sci.*, vol. NS-29, no. 29, Dec. 1982, pp. 2024–2031.
7. Nichols, Donald K.; Price, William E.; Kolasinski, W. A.; Koga, R.; Pickel, James C.; Blandford, James T., Jr.; and Waskiewicz, A. E.: Trends in Parts Susceptibility to Single Event Upset From Heavy Ions. *IEEE Trans. Nucl. Sci.*, vol. NS-32, no. 6, Dec. 1985, pp. 4189–4194.
8. Glaser, R.; and Masson, G. M.: Transient Upset in Microprocessor Controllers. *Digest of the 11th Annual International Symposium on Fault Tolerant Computing*, IEEE Computer Soc. Press, 1981, pp. 165–167.

Table 1. Error Incidence

Charge level, pC	First-order errors	Second-order errors	First-order pin errors	Functional errors
0.5	1	0	0	0
1	2	0	0	0
2	9	0	5	7
3	38	10	23	17
4	52	14	30	24
5	64	15	34	29
6	72	17	37	29
7	76	19	41	29
8	77	22	42	29
9	79	23	43	29
Totals	470	120	255	193

Table 2. First-Order Errors by Functional Unit

Functional unit	First-order errors	Percent errors per injection
Watchdog	132	37.7
Multiplexer	0	0
Decoder	68	19.4
Control	70	20.0
Countdown	70	20.0
ALU	130	37.1

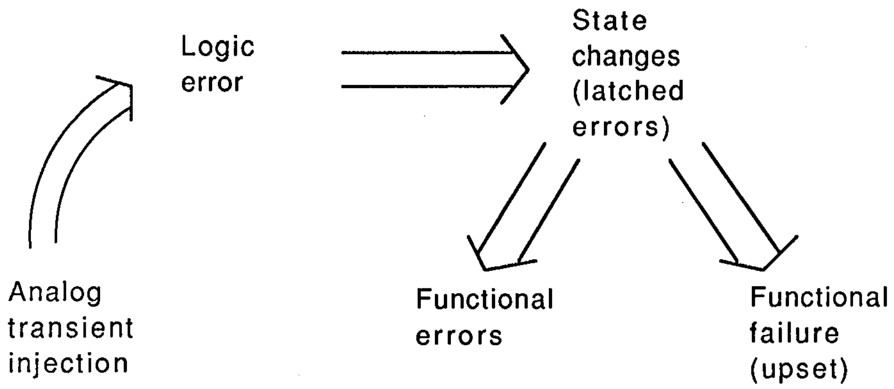
Logic error
State changes (latched errors)
Analog transient injection
Functional errors
Functional failure (upset)

Figure 1. Transient-injection process.

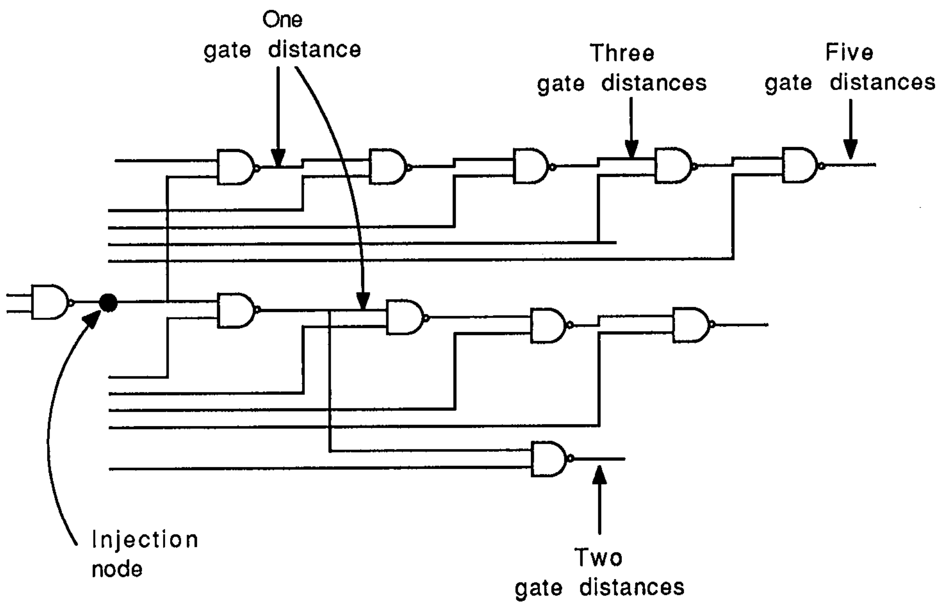

Figure 2. Gate distance from injection node in combinatorial circuit.

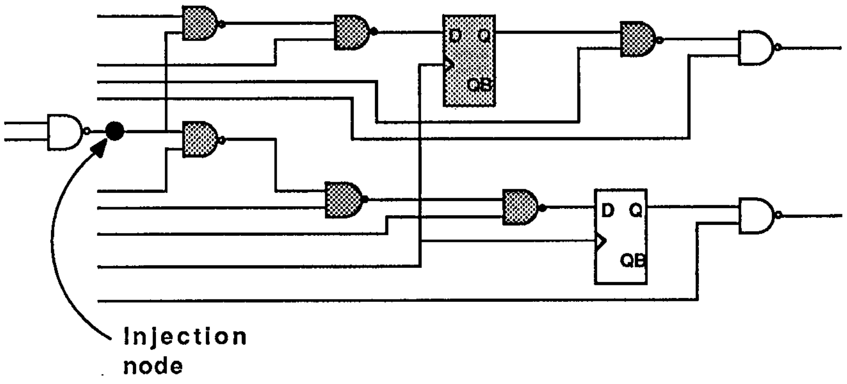

Figure 3. Gate distance in circuit including latches with shaded areas modeled at transitor level.

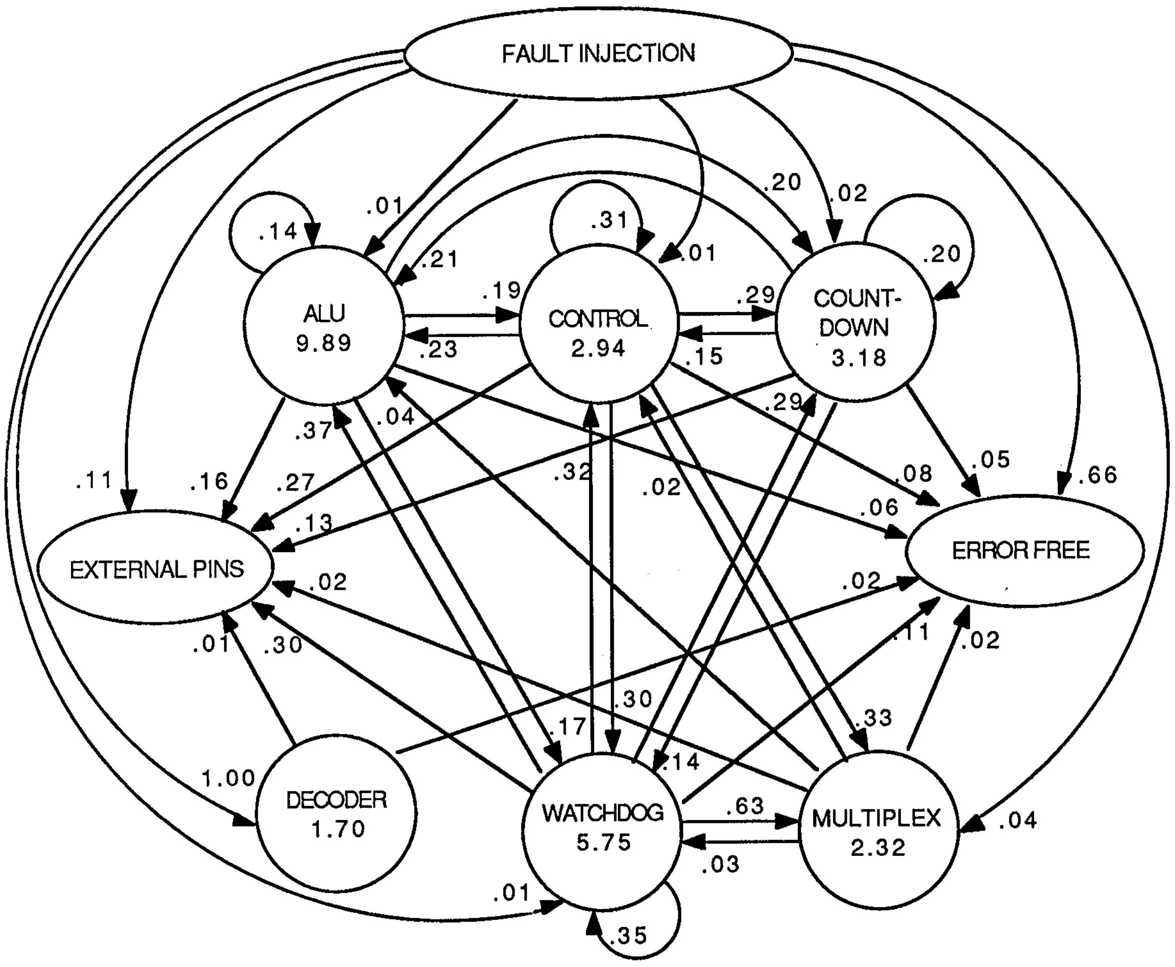

Figure 4. State transition model.

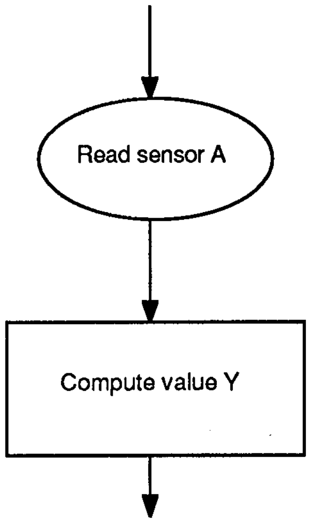

Figure 5. Flow diagram without fault tolerance.

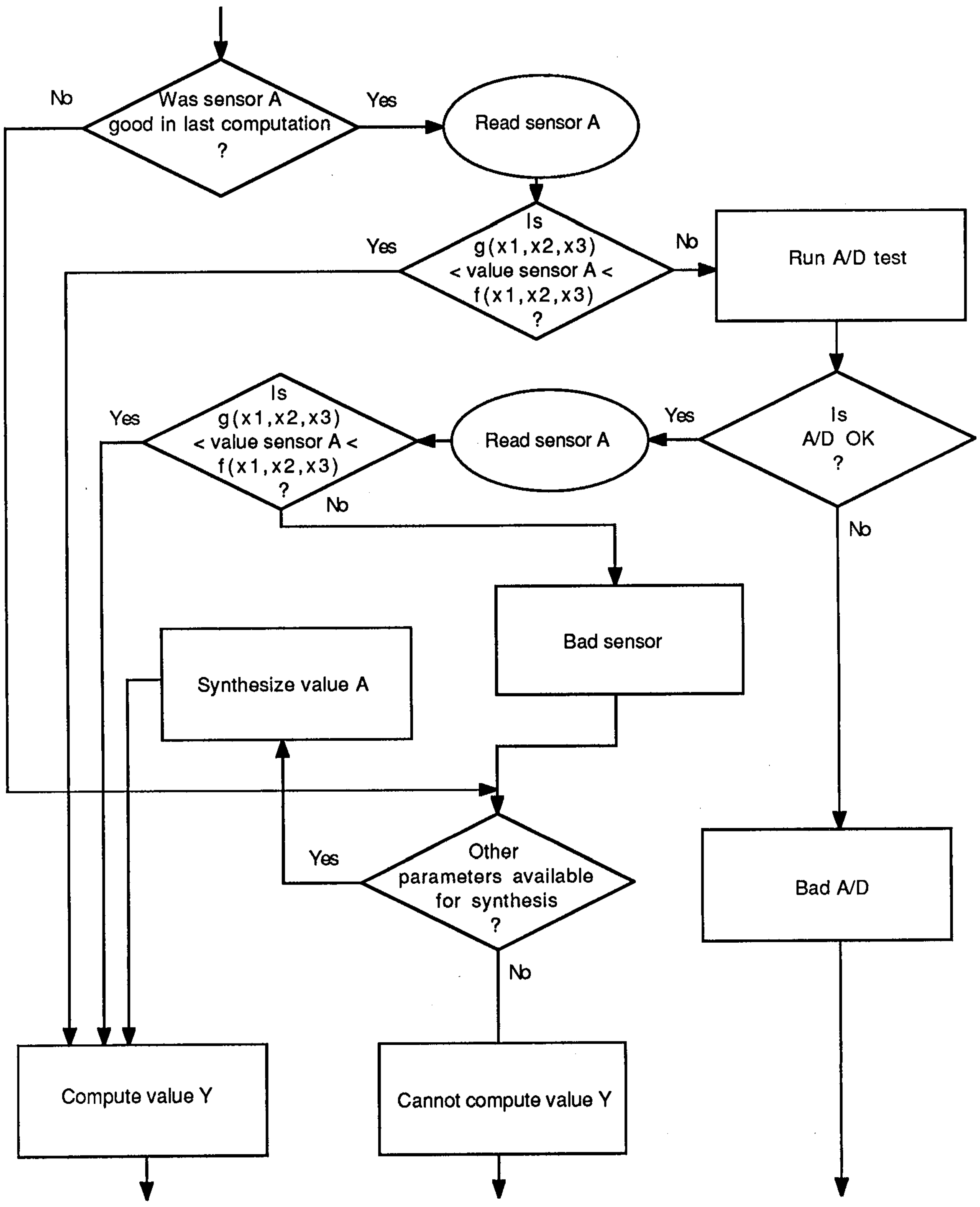
Was sensor A good in last computation ?
No
Yes
Read sensor A
Is g(x1,x2,x3) < value sensor A < f(x1,x2,x3) ?
Yes
No
Run A/D test
Is A/D OK ?
Yes
No
Read sensor A
Is g(x1,x2,x3) < value sensor A < f(x1,x2,x3) ?
Yes
No
Bad sensor
Synthesize value A
Other parameters available for synthesis ?
Yes
No
Bad A/D
Compute value Y
Cannot compute value Y

Figure 6. Flow diagram with built-in fault tolerance.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-4241	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Analog-Digital Simulation of Transient-Induced Logic Errors and Upset Susceptibility of an Advanced Control System		5. Report Date November 1990
		6. Performing Organization Code
7. Author(s) Victor A. Carreno, G. Choi, and R. K. Iyer		8. Performing Organization Report No. L-16714
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665-5225		10. Work Unit No. 505-66-21-04
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001		13. Type of Report and Period Covered Technical Memorandum
		14. Sponsoring Agency Code

15. Supplementary Notes
Victor A. Carreno: Langley Research Center, Hampton, Virginia.
G. Choi and R. K. Iyer: Computer Systems Group, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois.

16. Abstract
This paper describes a simulation to predict the susceptibility of an advanced avionics control system to electrical transients resulting in logic errors, latched errors, error propagation, and digital upset. The system is based on a custom-designed microprocessor, and it incorporates fault-tolerant techniques. The system being tested and the method of performing the transient-injection experiment are described. Results for 2100 transient injections are analyzed and classified according to charge level, type of error, and location of injection.

17. Key Words (Suggested by Authors(s)) Upset; Logic errors; Electrical transient; Analog-digital simulation	18. Distribution Statement Unclassified—Unlimited. Subject Category 62

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 11	22. Price A03