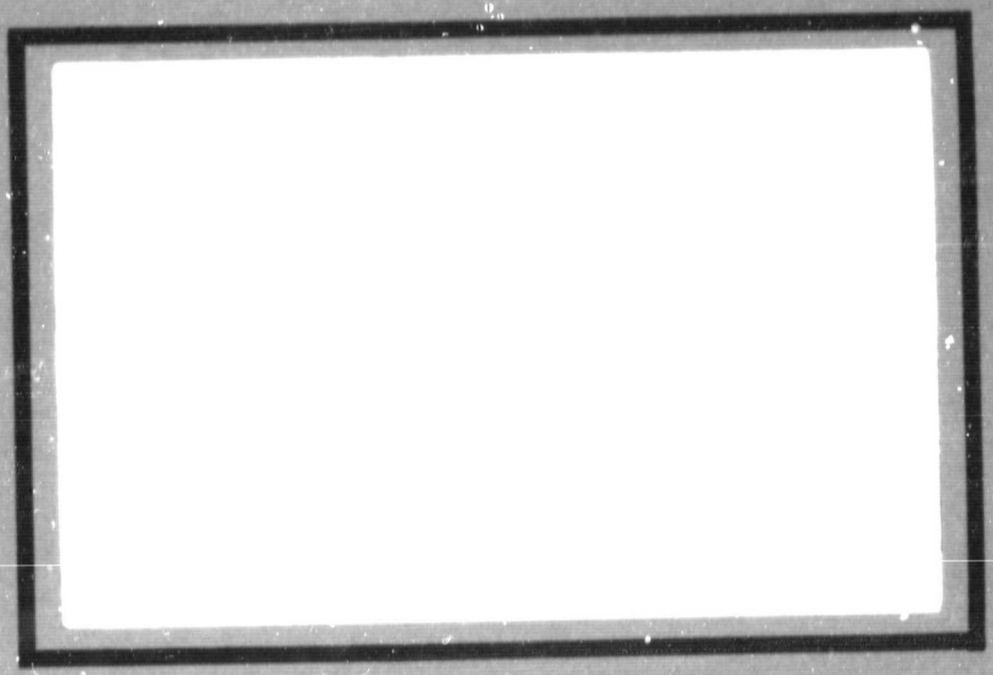

Virginia Polytechnic Institute and State University

Electrical Engineering

BLACKSBURG, VIRGINIA 24061

Status Report for NAG-1-174:

Chip Level Simulation of Fault Tolerant Computers

Performance Period:

June 1, 1981 Through May 31, 1982

Prepared By:

J. R. Armstrong and F. G. Gray

Electrical Engineering Department

Virginia Tech

Blacksburg, VA. 24061

Introduction

The purpose of this report is to give the status of grant NAG-1-174, Chip Level Simulation of Fault Tolerant Computers. The period of performance covered by the report is June 1, 1981 through May 31,1982. This particular document summarizes the work that has been performed during this period. The work is described in detail in three addendums to this report which are referenced below.

The work done under NAG-1-174 involves research into the use of chip level modeling techniques in the evaluation of fault tolerant systems. Modeling at the chip level involves modeling the internal device micro-operations as well as detailed interface timing, without employing a detailed gate level description of the device. Development of such techniques is important because of the enormous complexity of LSI devices, i.e. there can be tens of thousands of gates on a single chip. This complexity renders gate level modeling and simulation impractical for the long simulations required for system validation activities. An associated problem is that in many cases the detailed gate level model of an LSI device is known only to the manufacturer, who in general is unwilling to release this proprietary information. And finally, as we envision the digital systems of the future as being composed of many LSI devices interconnected in complex ways, it is important that we develop levels of representation that are accurate but

still envolve a manageable ammount of detail. The research carried out under NAG-1-174 is directed toward this goal.

During the performance period our efforts have been divided into four major areas:

1) development of chip level modeling techniques.
2) modeling of a fault tolerant computer (SIFT).
3) development of an efficient approach to functional fault simulation.
4) simulation software development.

Chip Level Modeling Techniques

Our experience in modeling on a previous Air Force sponsored contract [1] and that gained during the present work has resulted in the development of various techniques for modeling LSI devices. In order to preserve this information for others wishing to engage in chip level modeling we have prepared a document entitled "Chip Level Modeling Techniques". This document illustrates basic techniques for the modeling of the sequential and combinational logic aspects of LSI logic circuits. In addition, methods are presented for the accurate modeling of device interface timing, i.e. methods are given for modeling such input timing specifications as set up time, hold time and minimum pulse width. These basic techniques are then used to illustrate the modeling of devices peculiar to

microprocessor systems. The document represents our initial attempt to define for the unsophisticated device modeler techniques that he can use to model devices effectively. We plan to update the document on a continuing basis as new modeling techniques are developed, but we submit the first edition at this time.

Modeling of the SIFT Computer

As SIFT consists of a number of BDX-930 processors, modeling of the BDX-930 formed the first phase of the modeling procedure. The timing and control circuit and the CPU have been modeled completely at the present time. In doing the modeling, a great amount of effort had to be expended in defining the timing of the interface signals. Most of this timing information has been compiled from the timing specifications of individual chips. The timing information is listed in the report 'Modeling the BDX-930' which is also submitted as an addendum to this report.

The timing part of the timing and control board was modeled separately as 'BDXCLOCK' and the CPU board and the control circuits as 'BDXCPU'. The decision was prompted by the fact that the timing circuit, which takes care of extending microcycles, would have made the CPU model, which is largely synchronous in itself, more complicated. Also, this makes injection of faults in the timing circuits

easier. Notes on the modeling procedure used and the listings of the models are also given in 'Modeling the BDX-930'.

The BDXCLOCK model was tested out completely as soon as modeling was completed. Testing of the BDXCPU model has been, understandably, more involved. A temporary model was created for RAM memory which was used to store test programs. Models were also created for the bidirectional buses in the system - 'BDXDAT' and 'BDXACK'. The system was debugged and has been running successfully for sometime now. The CPU is being tested out instruction by instruction. At present, a third of the instruction set has been fully tested and the process is running smoothly.

Meanwhile, modeling of the actual BDX memory (with timings again being compiled from chip specifications) has been started. The I/O circuits will be modeled in the next performance period.

Throughout the modeling procedure, particular importance has been given to modeling so that fault injection is straight forward. This is especially so in the case of the CPU, where the internal architecture has been preserved intact. This makes it easy to translate a fault at the gate level into the model. Even a fault like one bit in the microprogram memory in the CPU being faulty should be easy to implement.

Simulation studies have indicated that the present BDX-930 model runs with a simulation efficiency of 24 clock

cycles per CPU second on an IBM 3032 processor.

Development of an Efficient Approach to Functional Fault Simulation

While we feel that the use of functional simulation is a necessity for LSI systems, it is true that the simulation of faults at the functional level is not as straight forward as the gate level fault simulation process. In gate level fault simulation, one merely causes a gate input or output to be stuck at one or stuck at zero and then simulates the fault. In using functional simulation, the fault insertion process is more complex in that if one is going to insert the fault internal to the chip, one must modify the functional model of the chip. Also, the nature of each such modification is peculiar to the particular fault being inserted. Thus the functional fault insertion process requires a detailed understanding of the operation of the device being modeled. There is also a basic question as to the time efficiency of the functional fault insertion process, i.e. how many faults per unit time can be inserted and fault runs made. It is true that in general a functional fault will cover a fairly large number of gate faults but one must still be concerned making the funtional fault insertion process as efficient as possible.

In response to these problems we've devoted

considerable effort during the performance period to development of functional fault insertion techniques and also to the development of sytem level approaches to allow efficient functional fault simulation. This work was carried out by Shirish Sathe, one of the graduate research assistants on the project. The results of this work are given in his master's thesis, "Functional Fault Simulaton in LSI Devices", which we are submitting as an addendum to this report. The thesis describes fault insertion techniques for the following types of faults: faulty micro-operations, timing faults, stuck at faults in internal device memory, interconnect faults (both stuck ats and shorts between lines), and transient faults. In addition to these fundamental techniques, the thesis describes a method for structuring the model to make the fault insertion process easier and to also allow a closer tie between real chip defects and functional faults. Also presented are methods for imbedding faults in models of good devices which can be invoked by means of external control signals to occur any time during a simulation run.

In order to perform the actual fault simulations efficiently, we have developed a software system which totally decouples the fault insertion process from the actual simulation process. The user prepares faulty models in one environment and then can submit a whole series of fault runs as a single entity. The complete series of fault runs will run to completion without operator intervention

with specified simulation data being recorded for each run.

Several other features have also been developed to aid in the fault insertion process. In the GSP simulation system, the interconnect between simulated chips is contained in a command file. An efficient method of inserting interconnect faults is to modify the good command file to create one modeling the faulty interconnect. An automatic technique has been developed for the preparation of these files. The user need only specify the module in the system at whose interface he wishes to insert faults. A program will automatically create an interconnect file for each stuck at fault for the specified chip, thus releaving the user of an error prone editing task, and saving considerable time.

Another feature has been added to facilitate the modeling of timing problems. Timing faults can of course be inserted by modification of the delay control parameters in the individual models as part of the fault insertion process described above. In addition to this however, a timing jitter option has been added which can be invoked during the actual simulation. With this feature, a pseudoramdom bias of a specified range, e.g. ±20% ,is added to each scheduled signal event, thus allowing the testing of system timing margins.

All of the features discussed above are described in detail in Mr. Sathe's thesis. The sum total of this work provides tools which will be of great value in carrying out

a meaningful and efficient functional fault simulation process.

Simulation Software Development

During the past year, a number of important software development tasks have been carried out as an adjunct to our basic research activities. First, the GSP simulation system has been installed on the Cyber 173 computer at NASA-Langley and a VAX 11-780 at Va. Tech. This is of course in addition to its normal operation on the IBM system at Va. Tech. Secondly, we have made modifications to the human interface of the system in order to make it easier to use. Finally, during the later part of the performance period we began the conversion of the BDX-930 cross-assembler and linking loader to allow it to run on the VAX 11-780.

Publications

Duing the performance period the following papers were accepted, presented or submitted for publication:

1) J.R. Armstrong and D. E. Devlin, "GSP: A Logic Simulator for LSI", Proceedings of the Eighteenth Design Automation Conference, pp. 518-524.

2) V. Puthenpurayil and J.R. Armstrong, "Functional Level

Modeling of LSI Devices", Proceedings of the Fourteenth Southeastern Symposium on System Theory, pp. 290-293.

3) S.Sathe,J.R. Armstrong, and F.G. Gray, "Functional Level Fault Simulation Techniques", Proceedings of the Fourteenth Southeastern Symposium on System Theory, pp. 285-290.

4) J.R. Armstrong, "Chip Level Modeling and Simulation", submitted to the IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems.

5) J.R. Armstrong and F.G.Gray, "Fault Diagnosis in a Boolean n Cube Array of Microprocessors", IEEE Transactions on Computers, vol. c-30,no. 8,Aug. 1981, pp 587-590.

Work to be Performed During the Coming Year

During the coming year we plan to carry out the following tasks under NAG 1-174:

1) Completion and Validation of the SIFT Model.

During the first performance period, a GSP model was developed for the BDX-930 processor and memory. In the next year we will develop models for the remaining SIFT logic, i.e. the I/O logic and the Broadcast Transmitter and Receiver Logic. The total model will then be validated by simulating the execution of SIFT diagnostic and operational programs.

2) Fault Injection Experiments.

The fault injection system developed during the first performance period of NAG 1-174 will be used to run fault injection experiments on the SIFT model. These expriments will be carried out on a VAX-11-780 at Va. Tech in preparation for the eventual installation of the fault simulation system in the AIRLAB environment.

3) Fault Modeling Studies

As an adjunct to the fault injection experiments we will attempt to develop fault modeling techniques that tie internal chip defects to the fault model. In this effort we will review the current literature on LSI device failure modes. Next functional fault models will be developed for generic LSI logic structures, e.g. PLA's, gate arrays etc. Finally, how these faults effect device micro-operations will be determined.

4) Improvements to the GSP Simulation System

During the past year we have compiled a list of improvements we would like to see made in the GSP simulation system. The two main areas where we desire to make changes are in the human interface and the main simulation loop. In this later case we will look at alternate ways of performing queue and I/O processing in order to improve the simulator's efficiency.

5) A Directly Executable VAX 11-780 Version of GSP

The present version of GSP is written in FORTRAN in order to ensure portability between systems. Thus, present GSP simulations involve a number of levels of interpretation. A module description is coded in assembly-like language which is converted by an assembler to an interger "microcode file". This file is interpreted by the FORTRAN simulator which itself is compiled into host machine language. One obviously pays a speed penalty for these different levels of interpretation. During the next performance period, we will investigate the possibility of compiling GSP module descriptions directly into VAX-11-780 assembly language. The assembly language that we currently use is very much like PDP-11 assembly language so that the idea certainly appears feasable. The VAX-11-780 has become a very commonly employed machine today and such a version of GSP would enjoy great portability. Also, it would allow for very efficient simulation in the VAX 11-780 configuration proposed for AIRLAB. Should our study indicate the value of such a version of GSP, we would use whatever resources that were available to begin implementation of such a system.

References

[1] J.R. Armstrong and F.G. Gray,"Microprocessor Self-Test", Final Report for RADC Contract F30602-80-C-0200.