General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

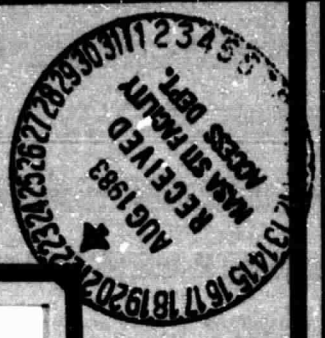
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

DRA



Virginia Polytechnic Institute and State University

Electrical Engineering

BLACKSBURG, VIRGINIA 24061

Final Report for NAG-1-174:

Chip Level Simulation of Fault Tolerant Computers

Performance Period
June 1,1981 - May 31,1983

J. R. Armstrong

Electrical Engineering Department
Virginia Tech
Blacksburg,Va. 24061

INTRODUCTION

The purpose of this report is to give the results of research carried out under Grant NAG-1-174, "Chip Level Simulation of Fault Tolerant Computers". The grant performance period was June 1,1981 through May 31,1983. This particular document summarizes the work that has been performed during this period. The work is described in detail in various papers and reports, copies of which have been transmitted to NASA along with this document.

BACKGROUND

The advent of LSI and VLSI circuits has led to increased use of functional level modeling and simulation. This is due to the tremendous complexity of these devices which causes great problems of scale for traditional gate level modeling and simulation techniques. LSI devices contain thousands of gates and gate level simulation of systems containing such devices can be prohibitively expensive. Moreover, the gate level models of these devices are usually known only to the manufacturer and he is generally 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 levels of representation are developed that are accurate but still involve a manageable amount of detail.

During the past four years at the Electrical Engineering Department of VPI, we have developed a form of functional simulation which we refer to as "chip level" simulation [1,2]. Modeling at the chip level involves modeling the internal device micro-operations as well as detailed interface timing without employing a gate level description of the device. This approach to modeling and simulation has been implemented in a system called GSP (General Simulation Program). GSP has been employed for device modeling research sponsored by the Naval Surface Weapons Center [3], the Rome Air Development Center [4], and the NASA Langley Research Center under NASA Grant NAG 1-174 and has been shown to be an effective system for the modeling and simulation of LSI devices.

In addition to the work outlined above, GSP will also be used to develop functional fault modeling techniques and a definition of functional fault coverage under a grant from IBM.

SUMMARY OF WORK CARRIED OUT UNDER NAG 1-174

During the performance period of NAG 1-174 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. (5) development of a high level language version of GSP possessing increased simulation efficiency. (6) development of a parallel architecture for functional simulation.

CHIP LEVEL MODELING TECHNIQUES

Our experience in modeling on a previous Air Force sponsored contract [4] 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, during the first year of NAG 1-174, we prepared a document entitled "Chip Level Modeling Techniques" [6]. 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 attempt to define for the unsophisticated device modeler techniques that he can use to model devices effectively. The original document was submitted to NASA as part of the first annual report for NAG 1-174 on May 31, 1982, but we have continued to make corrections and additions to it since then. A copy of the revised version accompanies this report.

In addition to the above report, a journal article was prepared covering material in the some area. The article, entitled "Chip Level Modeling and Simulation", will appear this coming October in the journal Simulation.

During the second year of NAG 1-174, we attempted to define the generic nature of a good functional model. In doing so, we developed a two layered approach to functional modeling. Figure 1 shows the two layered model structure. The top layer, the functional layer, reproduces the input/output behavior of the LSI device. This layer consists of a network of computational nodes which are interconnected in such a manner as to simulate the major signal paths in the modeled device. If one is interested in producing the behavior of the good device only, as is the case where the simulation is to be used for design verification or an automatic tester application, then this is the only layer that is necessary. However, if fault modeling is desired, then an additional layer is required to perform the mapping of physical faults onto the functional layer.

The concept of the fault mapping layer is built on the idea that in a chip level model of an LSI device, the micro-operations that comprise the model are themselves composed of micro-operations of smaller blocks of logic. To add the fault mapping layer, one codes the micro-operations of these blocks of logic in subroutine form. These subroutines are then called by the functional layer to implement the micro-operations of the chip level model. The fault injection process is thus simplified in that one need only determine the effect that fault has on the individual subroutines. In implementing our models in this fashion, we have found that it is better to represent these functional logic blocks, e.g. an ALU, as a group of subroutines as opposed to one large subroutine in order to alleviate the problem of passing a large number of parameters. The use of the fault mapping layer alleviates the problem of "micro-operation scatter" which troubled us in some of our early fault modeling efforts. This approach was used to model the BDX-930 processor. In this case, a group of subroutines was used to represent each LSI or MSI chip in the design. For the VLSI devices now being built, generic logic structures on these devices, such as PLAs, gate arrays, and programmable logic cells will be represented as subroutines.

This modeling structure is fully described in a paper entitled "Chip Level Modeling of LSI Devices" which was submitted to the IEEE Transactions on Computer Aided Design of Integrated Circuits and is currently in the review process.

Other systematic approaches to modeling are given in "Functional Level Modeling of Digital Devices", a masters thesis written by Venugopal Puthenpurayil, one of the graduate research assistants during the first year of the project. He developed the CPU/Timing model for the BDX-930 processor which is described in the next section.

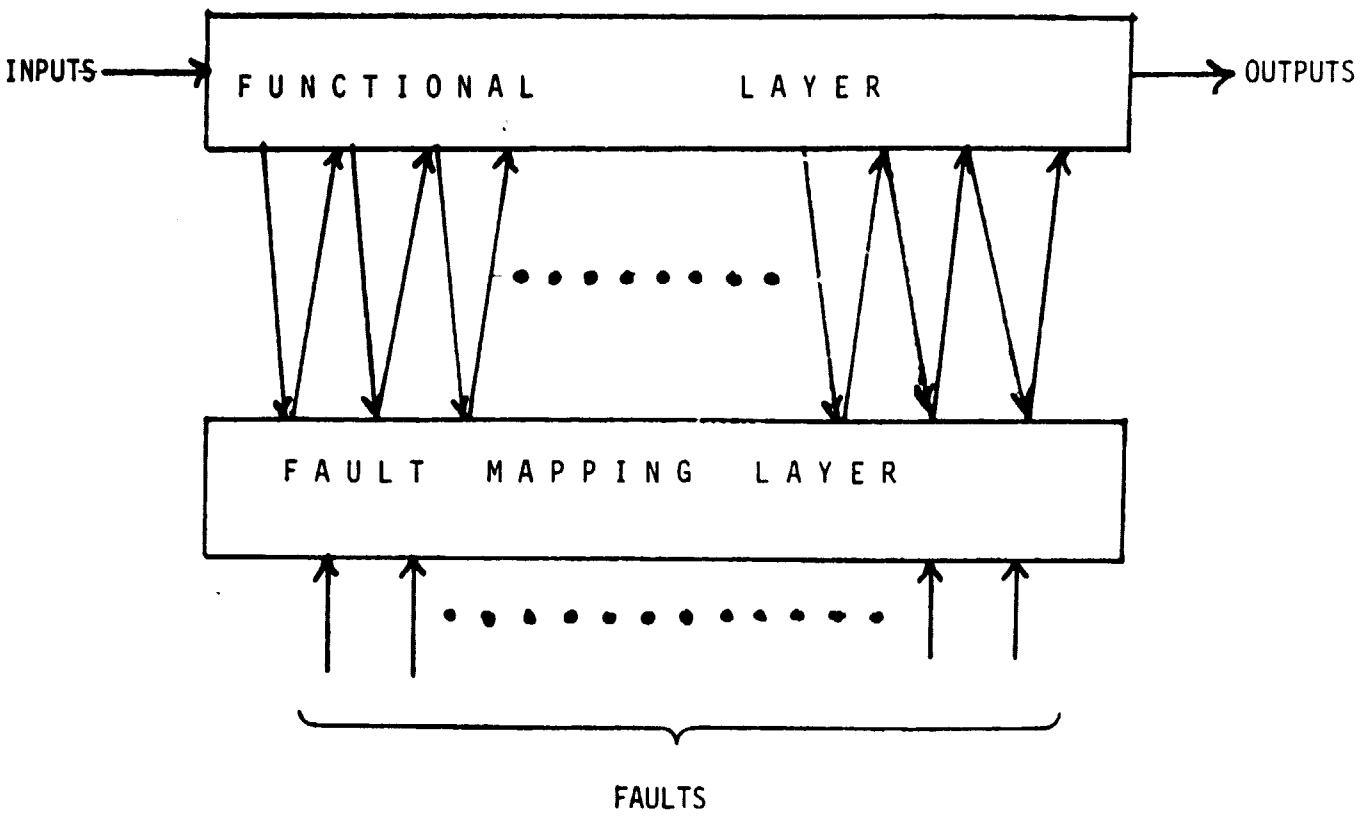


Figure 1

B) Modeling 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. Originally we broke the BDX-930 , which is a bit slice processor built from MSI and LSI devices, down into four functional models: processor, timing and control, memory, and broadcast and recieve logic. However as the models were completed and tested together, it became apparent that having the timing and control circuit model separate created a great deal of time queue activity which was unnecessary in terms of the goals of the simulation and which also greatly effected the efficiency of the simulation. Because of this, we merged the timing and control circuit with the CPU model. The effect of this and other improvements to the CPU model was to effectively double the efficiency of the model. In addition to this, the VA. Tech Computer Center recently installed an IBM 3081 dual processor sytem which is a 5 MIP machine. The overall effect of this was to make the CPU model run at an efficiency of 100 processor clock pulses per host CPU second.

The present status of the SIFT modeling effort is as follows: All models have been coded and checked out. The CPU/Timing and Control model and the Broadcast and Receive Logic model, in particular, were subjected to three months of intensive testing.

In doing the modeling described above, a great amount of effort had to be expended in defining the timing of the interface signals. Most of this timing information was compiled from the timing specifications of individual chips. The timing information for the CPU and timing and control logic is given in the report "Modeling the BDX-930" which was previously submitted to NASA. Additonal information is also contained in Venugopal's thesis (described in the previous section).

Throughout the modeling process, particular importance was given to modeling so that fault injection was straight forward. This was especially so in the case of the CPU, where a carefully constructed two layer model was developed (see previous discussion).

The modeling process was completed with approximately 5 months remaining on the grant period. The decision was made to apply the remaining resources to the development of the high level version of GSP and the parallel architecture for functional simulation(described below). As a result no system level fault simulations were performed. However the models are available and should NASA desire to resume this effort we could do so.

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. The two layer model discussed in the previous section should ease this process. However there is an additional problem that must be addressed. The variety of fault types, e.g. micro-operation faults, register stuck at faults, control faults, timing faults etc., dictates a sophisticated approach to the insertion process. This brings in to question the basic 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 with 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 system 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 for LSI Devices", which we have previously submitted to NASA. 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 addiion 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. (The fault mapping layer described above.) 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 submitt 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 devloped 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 for carrying out a meaningful and efficient functional fault simulation process.

SIMULATION SOFTWARE DEVELOPMENT

During the grant period, a number of important software development tasks have been carried out as an adjunct to our basic research activities. First, the GSP simulation system was 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. In particular, better mnemonics were chosen for existing modeling language instructions and several new instructions were added to account for some newly discovered modeling situations. These changes have been reflected in a updated GSP user's guide which has been previously transmitted to NASA.

During the second year of NAG-1-174, we converted the BDX-930 cross-assembler and linking loader to allow it to run on the VAX 11-780.

The conversion activities relating to the VAX 11-780 were considered important in light of the proposed AIRLAB environment.

A recent activitiy, actually begun after the termination of the grant period, involves the conversion of the present version of GSP to run on a personal computer with a relatively large memory space. It is expected that this conversion process will be completed this fall.

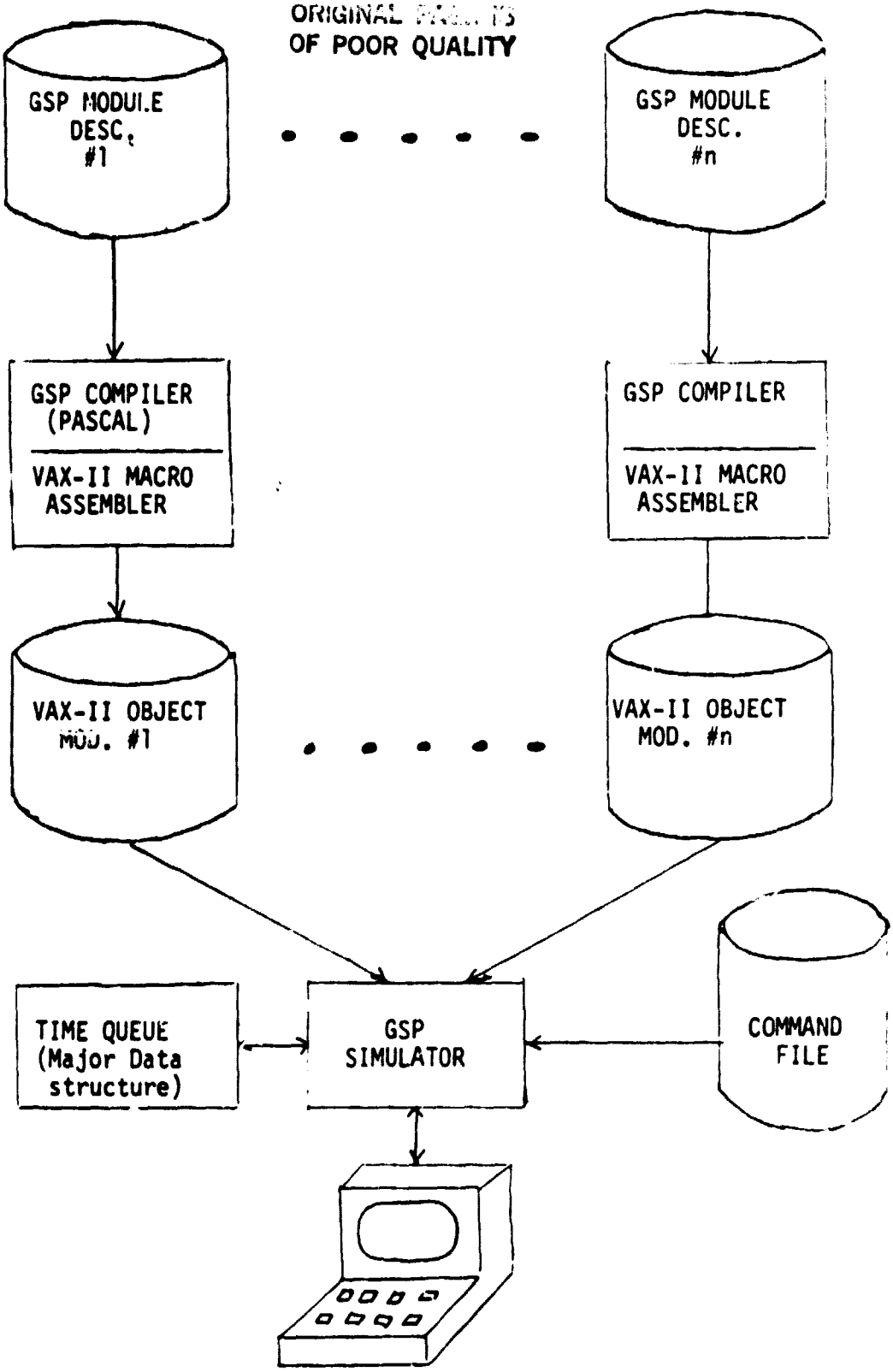
DEVELOPMENT OF A MORE EFFICIENT, HIGH LEVEL VERSION OF GSP

The current version of GSP runs too slowly for many applications, such as a complete simulation of the SIFT executive software. In order to improve simulation speed, hopefully by a factor of 100 or greater, the simulator was redesigned to be directly executable on the DEC VAX-11. In the current version, the hardware description language is translated into an integer microcode language, which is interpreted at run-time by the FORTRAN simulator program. This intermediate translation was eliminated by compiling the module description directly into VAX-11 object code routines executed by the assembly-coded simulator. This should provide a significant speedup.

The modeling language was also redesigned to be more powerful and support structured programming techniques. The assembly-like language was replaced with a high-level block structured language with constructs such as IF-THEN-ELSE, CASE, and arithmetic expression assignment. Many boolean functions and special functions such as RISE, FALL, SETUP, and HOLD were provided to tailor the language to the functional modeling of digital devices.

Symbolic level debugging commands were added to examine registers and pins by their source language level names, and set breakpoints on register or pin transitions, source statement line numbers, and subroutine entry and exit. These commands were implemented in such a way that they do not hinder the speed of the simulator when they are not being used, i.e. the simulator will run in two modes: a slower 'debug' mode, and the fast 'production' (fault experiment, system design validation, etc.) mode. Commands were also be added to provide a concise yet useful execution trace output. A detailed specification for the new language is given in is given in the "User's Guide to GSP II", a copy of which has been transmitted along with this report. A block diagram of the new simulation system design is shown below. The GSP simulator block is written in VAX-11 assembler, assembled once, and linked into an executable image with the object modules for all modules in a system.

At the time of the writng of this report, the programming of GSP II is 70% complete. It is expected that initial use of the system will begin this fall. This version of GSP should



SIMULATION SYSTEM BLOCK DIAGRAM

Figure 2

be of great value to the modeling and simulation activities associated with AIRLAB.

A PARALLEL ARCHITECTURE FOR FUNCTIONAL SIMULATION

The functional simulation process has two attributes which make it a suitable candidate for parallel processing. First the fact that LSI devices are represented by individual procedures gives a natural partitioning to the process. Secondly, a logic simulation process, once initiated, requires no input/output and thus should not be bandwidth limited by I/O rates as was ILLIAC [10]. A major problem that must be solved, however is how to solve the contention problem associated with access to the common time queue by the module proceedures and the accompanying synchronization requirements. A possible approach to the solution of these problems is outlined below.

To date, two announced approaches to parallel logic simulation have come from IBM and Bell Laboratories. IBM's "simulation engine" uses an array processing approach to perform gate level simulation in parallel [8]. It can perform either zero or unit delay simulation. The Bell Lab's proposal (in contrast to IBM's mechanization it has never been built), envolves the use of a pipelined organization to exploit parallelism [9]. It can potentially perform both gate level and functional level simulation. They propose that each stage of the pipeline use high speed logic and be micro-coded in order to decrease stage delay. The disadvantage of the pipeline approach is that the theoretical speed up is limited by the number of stages that process can be broken down into i.e. 5 or 6 in the case of logic simulation. Operational speed is of course reduced also by having very fast stages.

In contrast to the above, we have begun development of an approach that would employ a Multiple Instruction Multiple Data Stream (MIMD) architecture to peform functional logic simulation in parallel. We are attempting to exploit two sources of potential parallelism in the functional simulation process. First, within the simulation of a given system, be it "good" or "faulty", one can attempt to execute the proceedures representing the LSI devices in parallel. A second and perhaps greater source of parallelism is in the concurrent execution of N different systems, each having a different fault injected.

We have approached the problem by first developing a theoretical, process level model of the system. Next performance evaluation techniques have been applied to determine the theoretical through-put of the proposed system.

One the major problems with any parallel approach to simulation is the manner in which the time queue entries are handled. We have spent some time considering several approaches to this problem and we describe one promising approach here.

In functional simulation, the internal micro-operations and signal timing of the device are simulated. This simulation can be performed using activity oriented, event oriented, or process oriented time advancement mechanisms [7]. Process oriented simulation seems to offer the most promise of the three methods since it allows the designer to model each component module in isolation.The events internal to each module are kept isolated from the events occuring in other modules.

During the simulation, a time queue is associated with each module, containing the events that are to be executed by the module in time sequence.A master queue contains the inter-module communication events also in time order. A representation of these queues is shown in fig. 3.

The events to be executed in a module are grouped in tasks which represent a complete activity of the module. The events occuring in different modules are simulated in an arbitrary order, irrespective of the actual timing relationship as long as the order of interface events is properly represented.

During the execution of a task in a module, an interface signal may affect the completion of the task. However, for those signals that affect a task, one can predetermine the time frame during which they may appear [6]. Therefore the simulation sequence can be implemented as follows:

1. Estimate the time at which an interface communication signal can occur. Let it be Ta.

2. Simulate all events in the time queue of this module up to time Ta.

3. Suspend execution of the module and flag the exact point of suspension. Enter the task completion time in the master time queue.

4. If the expected interface signal has not occured until the task completion time, resume module simulation from the suspended point to the end of the task.

5. If the expected interface signal has occured in that time frame, enter that signal in the module time queue and resume execution again.

A block diagram of the previous simulation sequence is shown in Fig.4. The master queue execution and task sheduling pro-

MODULE n

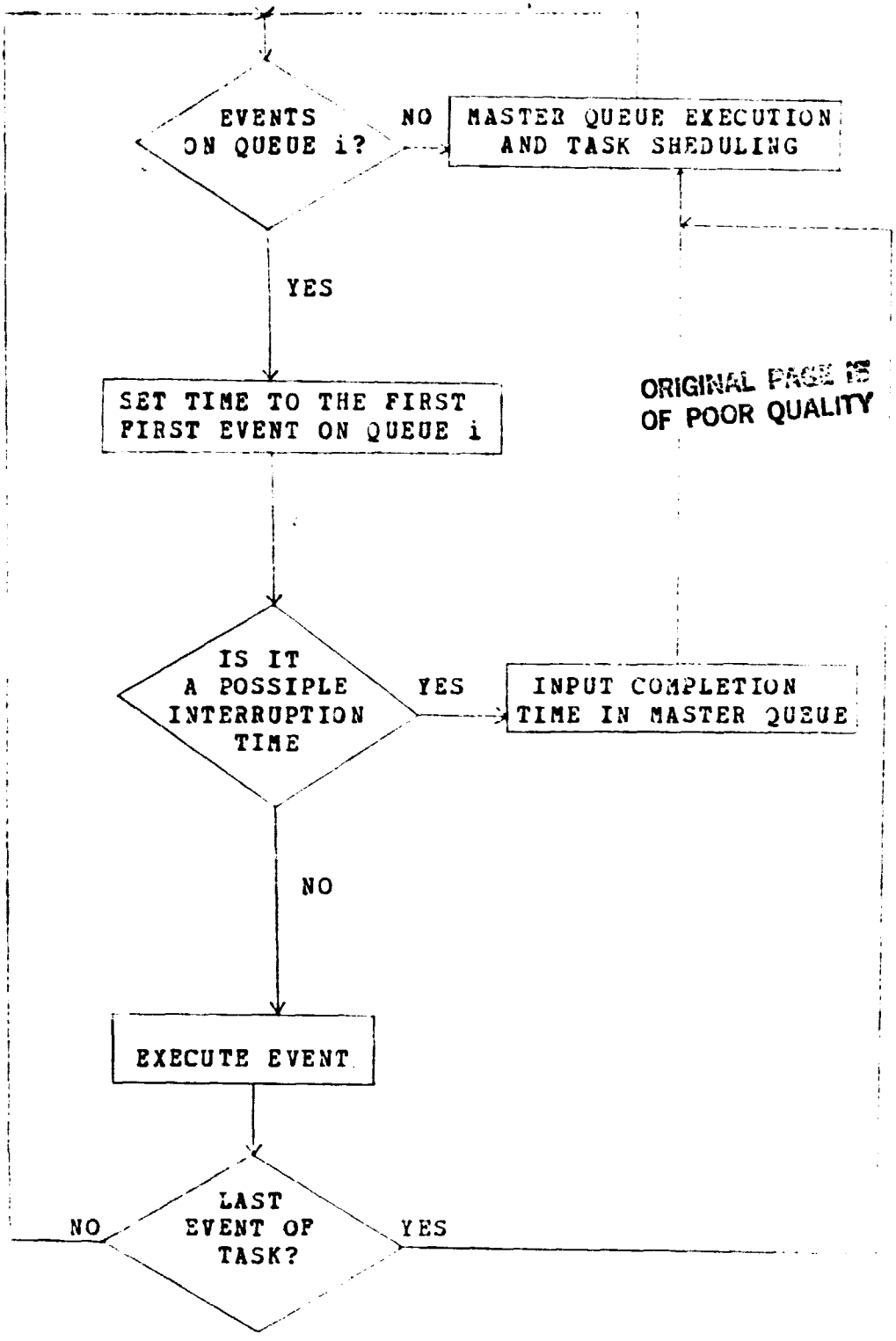
EVENT QUEUE

EVENT 11
EVENT 12
EVENT 13
POSSIBLE INTERRUPT
EVENT 14
END OF TASK 1
EVENT 21
. . .

MASTER QUEUE

COMMUNICATION EVENT 1
COMMUNICATION EVENT 2
COMPLETION TIME OF MODULE 1
COMMUNICATION EVENT 3
. . .

Figure 3



Simulation loop for module i

Figure 4

gram keeps track of the interface signals and propagates them to the corresponding modules. It also initiates task execution simulation sequences for the modules.

The proposed simulation scheme has the advantage of reducing alternate module execution at each time step, since only the intermodule communication signals affect the event execution at each module.

The details of this work are contained in a separate report: "A Parallel Approach to Functional, Process Oriented Simulation" which is in preparation at his time. A copy will be forwarded to NASA when it is completed.

PUBLICATIONS

During the performance period the following publications were prepared. The papers listed below were either presented at conferences or submitted to appropriate journals.

PAPERS

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", accepted for publication in Simulation, October 1983 issue.

5. J. R. Armstrong, "Chip Level Modeling of LSI Devices", submitted for publication in the "IEEE Transactions on Computer Aided Design of Integrated Circuits"

6. 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.

THESES AND REPORTS

1. V.Puthenpurayil, "Functional Level Modeling of Digital Devices", Masters Thesis,Department of Electrical Engineering, Va. Tech., Sept. 1982.

2. Shirish Sathe, "Functional Level Fault Simulation in LSI Devices." Masters Thesis, Department of Electrical Engineering, Va. Tech., June 1982.

3. J.R. Armstrong, "Chip Level Modeling Techniques", Technical Report 8124, EE Dept, Va. Tech, Nov. 1981.

4. D. Levlin,J. R. Armstrong,S. Sathe, and V. Puthenpurayil,"GSP User's Guide", Sept. 1982.

5. V. Puthenpurayil, "Modeling the BDX930", Technical Report, EE Dept, Va. Tech, June 1982.

6. J. R. Armstrong and F. G. Gray, "Status Report for NAG-1-174: Chip Level Simulation of Fault Tolerant Computers,Performance Period: June 1,1981 Through May 31,1982.

7. M. Iacoponi, "User's Guide for VAX Version of GSP and VAX BDX-930 RAM/ROM Loader"

8. J. Kerr, "GSP II User's Guide", EE Dept, Va. Tech., July 1983.

TEXTUAL REFERENCES

1. Armstrong, J.R., Woodruff, G.W., "Chip Level Simulation of Microprocessors", Computer, Vol. 13 ,No. 1, pp. 94-100,Jan. 1980.

2. Armstrong, J.R., Devlin D.E., "GSP-A Logic Simulator for LSI", Proceedings of the 18th Annual Design Automation Conference, pp. 518-524, Nashville, Tn, June 1981.

3. Armstrong, J.R., Thierbach, M. and Ellis, M.D., Final Report for NSWC Contract No N60921-78-A025, Vol. 3: Microprocessor Simulation.

4. Armstrong, J.R. and Gray F.G., "Microprocessor Self-Test", Final Report for RADC Contract F30602-80-200.

5. Armstrong, J.R. and Gray F.G.," Chip Level Simulation of Fault Tolerant Computers", Status Report For NASA Grant NAG-1-174, June 1, 1982.

6. Armstrong, J.R., "Chip Level Modeling Techniques", EE Dept. Technical Report, May 31,1982.

7. Leinwand, Sany M., "Process Oriented Logic Simulation, Proceedings of the 18th Design Automation Conference. Nashville Tenn., June 1981.

8. Fister, G.F., "The Yorktown Simulation Engine:Introduction" and Denneau, M. M., "The Yorktown Simulation Engine", Proceedings of the 19th Design Automation Conference, Las Vegas, Nev., June 1982.

9. M. Abromovici, et. all," A Logic Simulation Machine", Proceedings of the 19th Design Automation Conference, Las Vegas, Nev., June 1982.