intellitech

The Intelligent Use of Technology

(2)

COMPUTER-AIDED ENGINEERING (CAE) TOOLS FOR SPACECRAFT MULTIPROCESSOR DESIGN AND VERIFICATION

AN EXECUTIVE SUMMARY

INT-85-11

COMPUTER-AIDED ENGINEERING (CAE) TOOLS FOR SPACECRAFT MULTIPROCESSOR DESIGN AND VERIFICATION

AN EXECUTIVE SUMMARY

MARCH 1985

Prepared By: Dr. Samy Mahmoud

INTELLITECH CANADA LIMITED

352 MacLaren Street
Ottawa, Ontario
K2P OM6

Government of Canada   Gouvernement du Canada

Department of Communications

DOC CONTRACTOR REPORT   DOC-CR-SP -85-003

DEPARTMENT OF COMMUNICATIONS - OTTAWA - CANADA

SPACE PROGRAM

TITLE: Computer-Aided Engineering (CAE) Tools for Spacecraft Multiprocessor Design and Verification - An Executive Summary

AUTHOR(S): Dr. S.A. Mahmoud
INTELLITECH CANADA LIMITED
352 MacLaren Street
Ottawa, Ontario
K2P 0M6

ISSUED BY CONTRACTOR AS REPORT NO: INT-85-11

PREPARED BY: Dr. S.A. Mahmoud

DEPARTMENT OF SUPPLY AND SERVICES CONTRACT NO: OER 83-05075

DOC SCIENTIFIC AUTHORITY: Michel Savoie
Communications Research Centre
Ottawa, Ontario

CLASSIFICATION: Unclassified

DATE: MARCH 1985

TABLE OF CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

1. INTRODUCTION AND BACKGROUND

Microprocessor-based computer systems are used on spacecraft to increase reliability, extend mission duration and provide the on-board intelligence needed to meet mission objectives. These systems range from simple redundant (dual-processor) systems to distributed fault-tolerant multiprocessor systems.

To reduce the cost and risk associated with the development of such hardware/software (HF/SW) systems, DOC in 1981 initiated a project to specify, design, implement and demonstrate an appropriate set of CAE (Computer Aided Engineering) tools for the design, simulation, test, and maintenance of spacecraft on-board processing systems. The tools consist of a set of general purpose computer programs that enable system functional requirements (specifications etc.) and system HW/SW designs to be expressed in machine-processable form in order to facilitate the simulation and performance evaluation of alternate designs and to provide a format mechanism for system HW/SW development, verification and validation.

Work completed by Intellitech Canada Ltd. on this project in FY 81/82 and 82/83 includes a survey of existing CAE tools and techniques and the specification, design and partial implementation of an appropriate integration set of CAE tools for spacecraft applications.

An early version of a Computer Aided Engineering software package known as N.mPc was procured from Dicar Corporation (Ohio, U.S.A.) and installed for testing purposes on the Unix environment of a PDP 11/23 processor. A more powerful version of N.mPc was later installed on a VAX/780 computer running under the VMS operating system environment in the Analysis and Simulation Laboratory (A & SL) of CRC. An upgrade of N.mPc, referred to as the N.2 version, has been announced by ENDOT Inc. (Ohio, U.S.A.) and is being made available to CRC under a new software maintenance agreement between CRC and ENDOT.

2. OBJECTIVES OF THIS RESEARCH PROJECT

The main objectives of the current phase (FY 83/84 & FY 84/85) of the project are summarized in the following:

(1) The procurement, installation and checkout of the latest version of the software package N.mPc and/or N.2 and its associated software development environment (e.g., source code, C compiler and other high level language compilers and assemblers) needed to perform the tasks of this phase of the study. All software is to be installed and run on the VAX 11/780 under VMS in the CRC A & SL.

(2) The modification and extension of the N.mPc and/or N.2 library routines etc., to include the hardware models and software development facilities for selected microprocesors (e.g., F100-L and SBP 9989 and support chips etc.) used for spacecraft applications. Particular emphasis will be placed on the implementation of the SBP 9989 processor on N.mPc. The SBP 9989 hardware architecture and instruction set are to be assessed along with the investigation of the bus transfer mechanisms, interrupt linkages and interbus transfer mechanisms.

(3) Validation of N.mPc as a hardware simulation facility and as a microprocessor software development environment. The validation task involves the development of target microprocessor software in N.mPc. The software is then cross compiled and run on a simulation of the target hardware (itself running on the VAX 11-780).

(4) To specify, design and implement (using N.mPc and/or N.2) simulations of selected spacecraft fault-tolerant multiprocessor system architectures. The systems implemented will feature multiple and self-checking processors communicating on redundant serial buses with appropriate centralized and/or distributed control. Fault detection, isolation and recovery algorithms will be designed, implemented, compared and assessed. Feasible fault tolerant architectures are examined through mathematical reliability analysis in order to narrow down the selection of feasible architectures which can then be studied and simulated in detail. The main objective of this work is to develop a testbed for alternative fault tolerant architectures and operating system structures to be used in the simulation and development of future spacecraft on-board processing systems.

3. CONTRACT ACCOMPLISHMENTS

The following is a summary of contract accomplishments:

- The technical specifications and the methodology for using the N.mPc package have been fully documented following the installation of the package on the VAX/780 computer system at CRC. A detailed system description document and a user manual have been prepared.

- The implementation of an SBP 9989 development system on N.mPc has been successfully accomplished and resulted in descriptions of the SBP 9900/9989 processors in the N.mPc hardware description language. As in all N.mPc activities, there is an extra benefit resulting from the careful study of the hardware to be implemented. In this case, the benefit is a thorough understanding of the working of the SBP 9989 microprocessor. Based upon that understanding, the SBP 9989 can be compared with other microprocessors that are likely to be used in space applications. Examples of such processors include the Ferranti F100-L and the CMOS version of the Intel 8086.

- A simulation of an Intel iSBC 86/12 single board computer was performed successfully. It included an 8086 CPU, a dual port RAM, a ROM, a programmable interrupt controller (PIC), a Multibus interface, a global memory and an I/O facility based on N.mPc´s "raw memory" feature. These modules were designed and thoroughly tested and debugged during the course of the work. The validation of N.mPc as a CAE tool for microprocessor simulations was performed by implementing a "Simple Attitude Control Algorithm" as a "C" program, which was successfully run on the simulated 86/12 single board computer as well as on the actual Intel SBC hardware. The validation not only established the reliability of the simulated hardware but also demonstrated the I/O capabilities of the 86/12 simulation. The development of the validation program in "C" demonstrated the potential of a high level software development path introduced as a result of this work.

- A fault tolerant multiprocessor architecture suitable for spacecraft on-board processing has been developed, simulated and analysed. The hardware architecture concept is similar to the concepts developed previously for aircraft applications, but has been modified appropriately to provide higher reliability over the relatively long spacecraft mission life. An operating system capable of detecting, isolating and recovering from fault conditions and errors has also been developed and integrated with the hardware architecture. The functionality of the operating system has been tested through simulation. As well, the correctness and completeness of the hardware architecture have been tested, along with some basic operating system modules, using N.mPc simulation.

4. REPORTS PRODUCED AND SUBMITTED UNDER THE CURRENT STUDY

(1) VAX 11/780 CAE Tools for Multiprocessor Simulation:

N.mPc User´s and Applications Manual and Installation Guide

(DOC-CR-SP-84-067)

(2) VAX 11/780 CAE Tools for Multiprocessor Simulation

N.mPc Detailed System Description

(DOC-CR-SP-84-040)

(3) Validation of N.mPc/N.2 Microprocessor Simulation

(DOC-CR-SP-84-041)

(4) Simulation of the SBP 9989 Microprocessor Using the Computer Aided Engineering Tool N.mPc on a VAX 11/780

(DOC-CR-SP-84-023)

(5) Design and Analysis of Fault Tolerant Architectures for Multi-microprocessor Systems

(DOC-CR-SP-84-051)

(6) Design and Implementation of a Fault Tolerant Multiprocessor Operating System

(DOC-CR-SP-84-050)

(7) Simulation of a Fault Tolerant Multiprocessor System

(DOC-CR-SP-85-004)

(8) Computer-Aided Engineering (CAE) Tools for Spacecraft Multiprocessor Design and Verification - An Executive Summary

(DOC-CR-SP-85-003)

5. DESIGN METHODOLOGY USING CAE TOOLS

Interest in multiprocessor and distributed intelligence computer systems has increased dramatically during recent years. This interest has been generated by the demand for and the availability of multiprocessors with ever increasing performance/price ratios. However, the design of such systems is a fairly complex process which involves several alternative structures and determination of many system parameters. Control of these systems is not well understood, particularly for applications requiring high degree of fault tolerance and involving large number of processors. In addition ad-hoc techniques for designing large multiprocessor systems are not very advanced.

Design tools are needed which can be employed in the design and development of these systems and which can provide insights into the structure and attributes of alternative configurations. To meet these needs, a design and simulation environment for multiprocessor and distributed systems should have the following attributes:

- allow simulation of multiprocessor systems and at the same time allow modelling of such systems over a wide range of functional levels.
- allow changes to structure and topology of processor elements with minimum expense.
- provide monitoring and control facilities which can be employed anywhere in the target system.
- perform well when evaluating heterogeneous target architectures with large number of processors.
- allow the testing of different operating system structures and fault tolerance mechanisms.

Currently, no custom prototyping environment which possesses all of the above attributes is commercially available. In response to this perceived need, a design and simulation environment, N.mPc, has been developed and made available on a commercial basis. Other tools to serve the same purpose are currently undergoing development and will be available commercially in the near future.

N.mPc consists of six components used to either describe the hardware and software components of a microprocessor network, or to execute the simulation of a network. Figure 1 illustrates the components of N.mPc and their interactions.

Meta-micro, a generalized assembler along with the linking loader are used to generate the software executed by the simulated hardware components of the microprocessor network. Both are driven by a description of the target machine and instruction set, and are adaptable to generate code for either vertically or horizontally programmed machines. This linking loader produces a locatable code which is executed by a simulated processor or by an actual machine. The ISP´ compiler is used to produce simulation modules for individual processors and other hardware components of a network. The input language of the compiler is an extension of the computer hardware description language (CHDL). The ISP´ language allows specification of states for the implementation of processor registers and flags, memories for the simulation of memory, and ports which allow input to and output from simulated hardware.

The N.mPc ecologist and a simulated memory processor link the ISP´ processor modules with linking loader outputs to form complete network simulations. A run-time package is used to execute a simulation and allow extensive user interaction with the simulation.

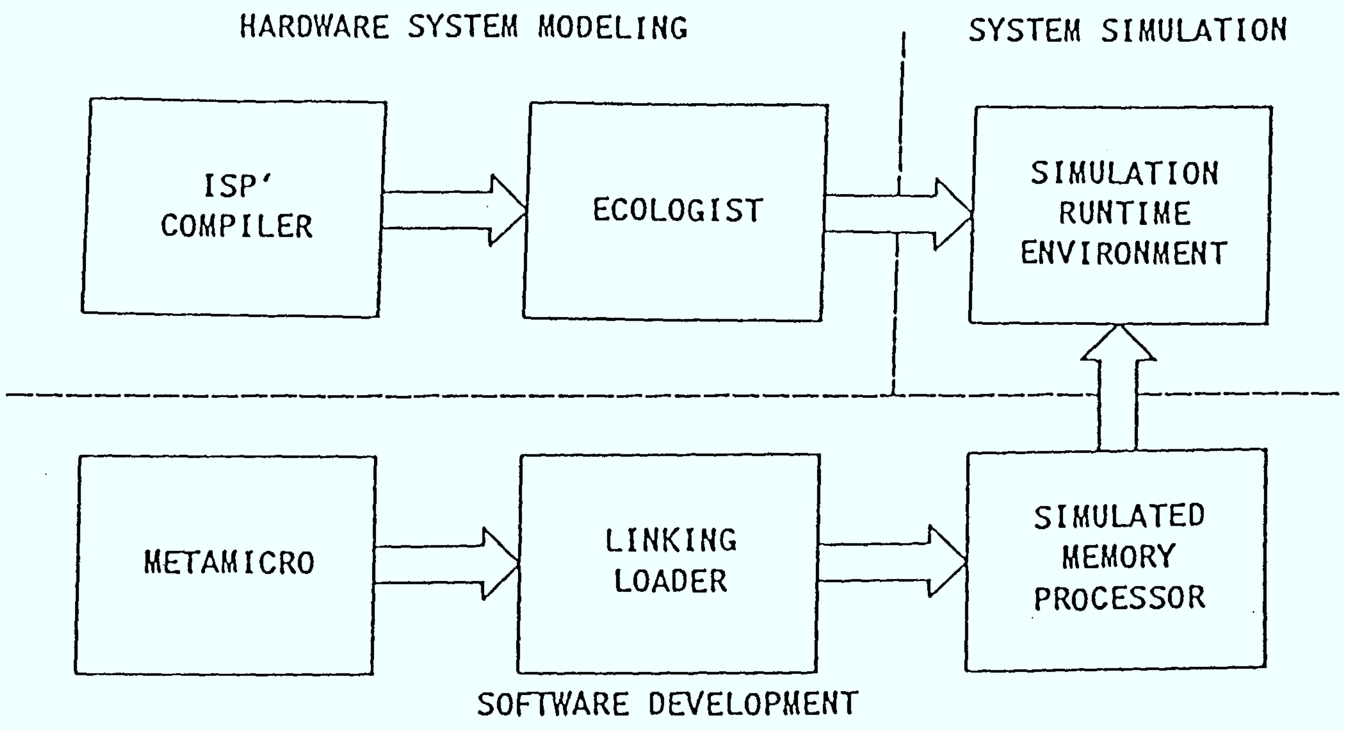

Figure 1 _ N.mPc Block Diagram

An N.mPc simulation might be viewed as going through several phases before finally reaching a design conclusion. The major phases are, simulation preparation, simulation execution, and post simulation evaluation.

Ultimately, the value of N.mPc will be measured in terms of its ability to generate results at a sufficiently low cost to justify its use in the design process. Cost, however, is a multi-dimensional variable containing components such as engineering cost, labour cost, and time spent. A realistic evaluation of N.mPc must address this issue in terms of cost versus return across a wide range of design situations.

These situations include the utility of N.mPc in the evaluation of general purpose processors, the design and simulation of special purpose processors and the design and simulation of microprocessor networks.

(i) Evaluation of General Purpose Processors Using N.mPc

With any general purpose processor, it is important to have a balanced functionally complete set of instructions and addressing modes. Many times, the instruction set of a new processor is planned long before any implementation problems are tackled. By writting a high level ISP′ description of the device, N.mPc makes it possible to construct an instruction executor which will allow the instruction set to be evaluated, without the need for extensive specification of the processors internal construction. Testing and modification can continue at this level until a suitable instruction set is developed for implementation. At this point there would certainly exist a general register set specification, and instruction bit mapping. Once the internal details of the processor are defined, the ISP′ description of the processor would slowly move from a high level algorithmic level to a register transfer level. Eventually even specified peripheral devices could be modelled to interact with this still unimplemented processor.

In addition to the above, a software gap usually develops for new processors as they appear on the market. To attack this common problem head on, N.mPc could be used to assist the movement of currently popular mini-computer operating systems and the high level languages to these processors. In such transfers, problems are usually encountered in the software/hardware operating system interface. N.mPc simulations of these interfaces such as device drivers and memory management control, could reduce the debugging time significantly.

(ii) <u>The Design and Simulation of Special Purpose Processors Using N.mPc</u>

The N.mPc system is equipped with the tools and primitives necessary to deal with the bit slice devices commonly used to construct special purpose computing machines. These machines are usually micro-programmed. Because micro-program instructions are usually more complex than standard machine instructions and because next instruction calculation is usually more complicated than the usual increment/branch, special attention has been given to micro program development. Metamicro and linking loader have several instructions expressly designed to aid in the development of software for horizontal architectures. Metamicro contains a register transfers syntax which may be used to clearly show the flow of information in register sets.

In both the general and specific purpose design, N.mPc can be used to evaluate the issues that complicate the design process, thereby reducing the need for prototype hardware, often a costly portion of a new architecture evaluation.

(iii) Simulation of Microprocessor Networks Using N.mPc

Due to the rapid growth of microprocessor technology, formal tools to describe and predict the behaviour of multi-micro processor systems are not yet available. Simulation is the only real alternative to prototyping. The prototype environment for multiprocessor systems is very harsh, usually requiring large investments of time and capital while often producing very little in the way of useable results. With the existence of an N.mPc library for the microprocessor family of interest, the user need only create the network topology description to have a complete hardware description of the network. The network software would have to be written, but because the simulation software can be used in the physical hardware, the user will have no additional software costs due to N.mPc. Even without suitable tools to approach multiprocessor designs, networks have generated widespread interest. Designs, for example, may offer high processing power versus cost performance, or high levels of fault tolerance. Some of the new 16 and 32 bit microprocessors have special software and hardware features to support multiple processor configurations.

With a simulation system such as N.mPc, the user may solve the problems encountered in current multiprocessor configurations, as well as acquire the data to construct higher performance microprocessor systems.

The experience gained in working with N.mPc as a simulation tool throughout this study indicated, however, the existence of some limitations which affect its utility in simulating complex systems such as the fault tolerant architecture and its associated fault tolerant operating system. These limitations are described later in this report.

6. GENERAL DESCRIPTION OF THE FAULT TOLERANT MULTIPROCESSOR SYSTEM

The hardware architecture of the fault tolerant system consists of three separate components: the central control system, the peripheral network and the peripherals. The central control system consists of the Central Processors (CPs) and the interconnection network connecting them. The central processors communicate with each other via dedicated channels. They constitute the highest on-board processing authority and handle all the processing and control requirements of the satellite.

The function of the peripheral network is to provide a fault tolerant connection between the central processors and the devices they control. The network consists of a serial redundant bus to which all the peripheral devices are connected. The central processors do not have direct access to the redundant bus. Instead they communicate with each other via a set of dedicated Interface Processors (IPs). The function of the IPs is to handle the low level details and the protocol of the peripheral network and to provide a fault tolerant interface between the asynchronous central processors and the synchronous redundant bus.

The interface processors and the peripheral devices are connected to the redundant bus through specially designed gates that ensure a device that fails will not incapacitate the system. Also the gates provide the means for the isolation of any faulty device from the system.

The system hardware described must be complemented with suitable software in order to fulfill its intended purpose. The interface processors perform relatively simple and repetitive tasks. Therefore their software is straightforward and can be though of as firmware. Their main function is to pass data between the central processors and the peripherals in a controlled and fault tolerant manner. This activity is directed and controlled by the central processors.

The operating system resides and runs on the central processors. It provides the environment for the application tasks to run and handles the masking and isolation of faults. In addition, it is responsible for the recognition of hardware and software failures and the reconfiguration of the system around them.

The proposed fault tolerant architecture has many common features with a number of existing fault tolerant architectures for aircraft systems. The two most important of these are SIFT, and FTMP. Many elements of the FTMP system were found to be particularly suitable for peripheral interfacing and were adopted for the peripheral network. These elements (serial redundant bus and gates) had to be modified in view of the radically different requirements of a satellite system as opposed to an aircraft system.

The main objective of operating system simulation is to make possible the testing of the completeness and correctness of the various functions of the operating system both under normal and faulty conditions, to the greatest extent possible, given the unavailability of the system hardware. To accomplish this a test module was developed and linked with the operating system code. This module emulates both the functions of the higher layers of the operating system (processor managers and global executive) and the functions of the hardware (interprocessor communications).

In addition, this module enables the user to interactively inject errors in the system and observe the system´s behaviour. At the present stage the system can detect, isolate and recover from most external errors including soft transmission errors, communication line failures, task failures and processor failures. All these external errors originate from within the central control system. Another class of errors which cannot be tested at the present time are the errors originating from the peripheral network. Any serious attempt with practically meaningful results to include these errors would require the availability of the system (target) hardware.

The interactive testbed has been designed as one self-contained program. The underlying hardware is simulated internally by moving data from one processor structure to another. Therefore it is entirely independent of the operating system under which it runs.

Originally the operating system simulation was developed on a VAX 11/750 running UNIX 4.2. Once the debugging was completed it was moved to the CRC VAX 11/780 and run under the VMS operating system. The operating system and testbed software developed are transportable to any host computer provided it has sufficient memory and a full C compiler. The transportability of the software to other host computers is possible since neither the testbed nor the operating system itself make use of any special features of UNIX.

The CAE tool N.mPc was used for functional testing of the fault tolerant multiprocessor hardware architecture by first developing descriptions of all necessary hardware modules in N.mPc's hardware description language ISP'. The next step was to describe an interconnection scheme for the simulated hardware components of the multiprocessor architecture. Then some test software for the programmable hardware modules (descriptions of the Intel 8086 microprocessor) were developed. Finally simulated hardware and test software for the microprocessors were integrated under N.mPc to form an executable simulation that allows functional testing of the fault tolerant multiprocessor architecture and of the behaviour of the multiprocessor system in response to certain failures. N.mPc was also used to test the integration of some operating system modules with the hardware architecture.

7. SUMMARY OF RESULTS AND CONCLUSIONS

A conceptual design for a fault-tolerant multiprocessor architecture has been analysed, simulated and its suitability for spacecraft on-board processing has been demonstrated. The multiprocessor architecture is particularly suited for the relatively long spacecraft mission duration.

The approach chosen for the design of the fault tolerant multiprocessor system is novel in the sense that fault tolerant features and supporting mechanisms are embedded in both the hardware architecture and the operating system software. The hardware has multiple redundant components that are controlled by fault detection mechanisms in order to prevent the propagation of errors from the faulty component to the remainder of the system. The operating system software contains all the intelligence needed to detect errors, identify their sources, take the necessary action to remove faulty units, reallocate the processing tasks and reconfigure the system to adapt to the new operational state.

One important limitation to the fault tolerant system developed in this study stems from the fully connected topology of the main processors. This topology means that the number of main processors must be limited in practice since (n-1) interface ports are needed for a configuration of n processors. This imposes an upper limit on the total processing power of the system and precludes the flexibility of further expansion into very large configurations. However the selection of the fully connected topology was made in favour of arriving at a provable and tractable fault tolerance capability and at the expense of the flexibility of constructing a topology with very large number of processors. This separates our system from many other published configurations in which the number of processors numbered in the hundreds. Finally it should be observed that our system is targetted for applications requiring finite processing power but high degree of fault tolerance and graceful degradation.

The operating system can detect and isolate a wide range of errors and failures. Simple transient errors are masked from the system whereas component and software failures are recognized, isolated and repaired to the greatest possible extent. This is accomplished by employing hardware and software redundancy along with the appropriate fault tolerant software to manage the redundant resources of the system.

Redundancy ensures the fault tolerant operation of the system but is achieved, however, at the cost of extra hardware resources and reduction in processing throughput. In order to ensure the timely and reliable isolation of faults any software running on the system must run in triplicate (or sometimes in more copies). In other words for every three processors in the system, the equivalent throughput of one processor is obtained. Besides the fault tolerant features of the software require extra processor resources for voting and reconfiguration. It is estimated that the overall throughput of the system will be 25% of that of a basic system without any fault tolerant features.

Although the system presented here was originally designed for spacecraft applications, it is not limited to such applications. In fact the proposed architecture could be used for any system requiring finite processing power with ensured continuous operation. The ability of the system to isolate the failures automatically and reconfigure itself around them makes it ideal for applications where the cost of maintenance and the logistics associated with it require minimum human interference. Examples of such systems include multi-microprocessors used in the operation, monitoring and maintenance of nuclear plants, in computer communication networks and in remote-site data collection and distribution (eg. environment and resource management applications).

The simulation approach to hardware and operating system software design proved to be useful in the course of this work. Several times it has been necessary to add a new feature to the guardian initialization mechanism, change the interconnection of some hardware modules or complete the description of a bus interface which the simulation proved to be inaccurate. All these changes would have been extremely time consuming and costly if they had to be done on a real prototype hardware. The inherent flexibility of the simulation approach allows us to evaluate several design alternatives of a hardware system in a short time. This allows the designer, when he finally commits himself to implementing a certain design, to rule out early conceived options that proved to be incorrect or inefficient through the simulation work.

Concerning the adequacy of N.mPc as a design tool for multiprocessor systems, several points can be made:

- In the course of this work the CAE tool N.mPc clearly proved to be useful as a hardware design and simulation tool. A complex multiprocessor system could be simulated and tested in a relatively short period of time.

- Frequent design changes to the simulated multiprocessor architecture showed the flexibility of N.mPc as a hardware design and simulation tool.

On the other hand, several limitations to N.mPc have been identified which tend to severely reduce its utility in the simulation and testing of fully integrated multiprocessor systems:

- N.mPc's slow execution speed results in a prohibitively high demand on the host CPU time if the test software modules are of substantial size. This was the case for the fault tolerant operating system.

- In the N.mPc simulation conducted here, the main (control) processors were represented each by Intel's 8086 processor. A full description of this processor is included in the library of N.mPc. The 8086 version within N.mPc was developed based on the available 8086 VLSI chip details. Like the case with many sophisticated processors, the commercially available VLSI description is not guaranteed to be complete nor absolutely accurate (bug free). All unidentified faulty attributes in the description will thus be propagated to any simulation which uses the library copy of the processor description. This complicates the process of tracing the sources of bugs when high level operating system software modules are tested in the simulation.

- The fact that N.mPc simulates the hardware down to the register transfer level is useful when newly designed hardware modules are being tested. However, when the focus of the simulation shifts to higher levels of structure modules, N.mPc still simulates every register transfer in every microprocessor involved in lower level instruction executions resulting in a large simulation overhead. This aspect was encountered when an initial attempt was made to run the relatively complex software of the fault tolerant operating system on the fault tolerant microprocessor architecture.

The next generation of CAE tools is expected to be endowed with top-down design and simulation features to allow the designer to follow a methodology in which he can test the lower level modules of the operating system down to register transfer level of details. Higher level modules can then be simulated and tested with the already tested lower levels replaced by macro instructions or functional blocks. This will enhance the simulation performance by several orders of magnitude over what is currently attainable by a tool like N.mPc. It will thus make it possible in practice to simulate a sophisticated system such as the fault tolerant architecture reported here in its full fledged configuration in a reasonable period of time and using modest computing resources.

A new version of N.mPc, called N.2, has been introduced by the vendor and is available on the VAX/VMS environment. The new version N.2 incorporates a few features aiming towards making the original N.mPc more powerful.

8. RECOMMENDED DIRECTIONS FOR FURTHER STUDIES AND RESEARCH

The study completed so far investigated the design, simulation and testing of the basic functions of a fault tolerant multiprocessor system. The results presented however, do not cover all potential capabilities that can be included in such a system. Extending the fault tolerant system to its full fault tolerant potential remains to be the subject of further research and development.

Further work on the subject may include the expansion of the higher layers of the operating system (the processor manager and the global executive). The processor manager can be expanded so that it can handle the creation, deletion and relocation of tasks. To do this some memory allocation and management software is required which will have to be developed and tested for a specific target hardware implementation.

The global executive can be expanded by using more intelligent reconfiguration techniques and by increasing the number and types of failures it can handle. These should include failures in the peripheral network as well as partial failures of processor modules. It should also be extended so that it can enforce a graceful degradation when module failures result in overall reduction of the collective processing power of the system. Graceful degradation would necessitate the removal of certain functions according to a priority scheme which specifies an order of importance. This will enable the system to continue performing its basic functions even as processing power continues to decrease with module failures.

Some of these extensions can be tested in a satisfactory way by using simulation techniques similar to those employed in this report. However, many of the functions can only be tested on the actual target system hardware. Therefore the realization of the hardware and the porting of the operating system to it are essential if this work is to be carried to a conclusion.

The system in its present state can handle only single errors (faults) occurring in serial order. The system should be expanded to handle concurrent errors (faults). Since the number of combinations of such errors is large, it would be difficult to anticipate their nature in advance. A fault tree analysis will be required which will benefit from the advances made in the field of expert systems in artificial intelligence.

Finally, it is essential to monitor the technology and the availability of the next generation of CAE tools. Future CAE tools should not only be able to do hardware simulations on the register transfer level but should also include the capability of simulating complex hardware modules as "black boxes". In this manner simulations could be moved to higher functional levels while minimizing the demand for computer time for subsequent simulations.

intellitech

Intellitech Canada Ltd
352 MacLaren Street,
Ottawa, Ontario
K2P 0M6
(613)235-5126