

CONF-850903--14

FAFTRCS - AN EXPERIMENT IN COMPUTERIZED
REACTOR SAFETY SYSTEMS*

CONF-850903--16
DE85 018417

by

G. H. Chisholm

EBR-II Division
Argonne National Laboratory
P. O. Box 2528
Idaho Falls, Idaho 83403-2528

The submitted manuscript has been authored by a contractor of the U. S. Government under contract No. W-31-109-ENG-38. Accordingly, the U. S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U. S. Government purposes.

Submitted for Presentation
at the
American Nuclear Society
International Topical Meeting
on

"Computer Applications for Nuclear Power Plant Operation and Control
September 8-12, 1985
Pasco, Washington

REPRODUCTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

*Work supported by the U.S. Department of Energy under Contract W-31-109-ENG-38

MASTER

ABSTRACT

Nuclear Power Plant availability and reliability could be improved by the integration of computers into the control environment. However, computer-based systems are historically viewed as being unreliable. This places a burden upon the designer to demonstrate adequate reliability and availability for the computer. The complexity associated with computers coupled with the manual nature of these demonstrations results in a high cost which typically has been justified for critical applications only. This paper investigates a methodology for automating this process and discusses a project which intends to apply this methodology to design verification and validation for a control system which will be installed and tested in an actual reactor control environment.

1. INTRODUCTION

1.1 The Myth of Fault-Tolerant Computers

A myriad of technologies must be combined to realize an ultra-reliable computer-based system. The visible aggregation of these technologies resides in the recent explosion in development of fault-tolerant computers.

A myth associated with fault-tolerant computer technology leads the uninitiated to conclude that a system based upon a fault-tolerant computer is inherently reliable. Unfortunately, a careless application of a fault-tolerant computer may easily defeat the best intentions of the computer designer and provide dismal reliability and availability.

Maintenance of the fault-tolerant design strategy[1] may be attained by a number of approaches. One approach is that the design team becomes cognizant of those properties of the computer which are essential for fault-tolerance and confirm that the final design will comply. A second approach is the development of a number of building blocks which are demonstrably compatible with the underlying fault-tolerant strategy and which are generic to many applications on the target system. Inherent in each of these approaches is the requirement for "demonstrably reliable" or "confirmation of reliability".

The disciplines involved in the implementation and confirmation of an ultra-reliable, computer-based system are varied. One of the most important aspects of coalescing a design team, whose combined expertise encompasses the breadth of the total design problem, is the development of inter-discipline communication.

Within the varied disciplines, there exists a dialogue about which approach is best suited to solution of the subset of the problem within their purview. This intra-discipline dialogue centers around the fact that the touted approaches have yet to be validated. A further complication arises from the inter-discipline dialogue which centers around which discipline best solves a subset of the total-system problem.

There is a need for a demonstration project which would endeavor to amass a team from diverse disciplines whose goal is implementation and confirmation of an ultra-reliable, computer-based system. This paper describes an ongoing project which

will attempt to demonstrate a total systems approach to the design of a reactor safety subsystem. The organization of the paper is: a brief overview of the project and a discussion of automated techniques for formal design verification.

2. FAFTRCS PHASE I

2.1 Background

ANL is currently investigating the issues associated with application of computers in critical systems applications. These investigations utilize available technologies as much as practical, i.e., fault-tolerant processors (FTP), software reliability models, Markov modeling. Novel techniques which could improve the existing methods also are to be considered. This project, "The Full-Authority, Fault-Tolerant Reliable Control System," has been given the acronym FAFTRCS. Full-Authority is the industry vernacular applied to a control/controlled system symbiosis where adequate functioning of the controlled system is totally dependent upon proper functioning of the controller. Fault-Tolerant refers to a real-time, multi-processor which will perform its design function in the presence of a failure. The ultimate goal of FAFTRCS is to design and demonstrate an automated reactor control system for which a prospective licensee could take safety credit during the licensing process. This goal demands the extension of technology and is the subject of current research which may not culminate for 5-7 years. To this end, the project has been subdivided into three phases. Each of the early phases is intended to demonstrate evolving technologies in support of the next phase.

One facet of the investigations considers reducing the cost of design verification and validation (V&V). The strategy being pursued is directed toward developing an automated methodology for design V&V where the machine performs the mundane tasks in order to free the person to accomplish more creative tasks. Automated reasoning is being applied in this area as a potential method of reducing the complexity associated with a formal analysis of a computer-based system. An important aspect of this research is to assure that the resulting analysis is presented in a format which is usable to the safety review community. As a measure of success, this work will be applied to an actual system which will be tested at the Experimental Breeder Reactor No. II.

The FAFTRCS project is concerned with total system reliability which incorporates the software, hardware and their respective interaction. The use of fault-tolerant hardware provides a significant improvement in reliability for the overall system. However, the properties of the design that are essential for fault-tolerance must be established. Though significant benefit is realized from the application of existing fault-tolerant designs, the FAFTRCS project will perform a formal design analysis of the hardware.

2.2 Description of FAFTRCS Project

Phase 1, Fault-Tolerant Flow Trip (FT2), is focusing on V&V of the generic issues for ultra-reliable computer-based control (i.e., fault-tolerant processor), developing automated aids for V&V, and application of signal validation software. Phase 1 will culminate with the installation of a fault-tolerant flow channel in the EBR-II Plant Protection System (PPS).

Phase 2, Fault-Tolerant Analytical Redundancy, will expand the software V&V methods developed during Phase 1. In addition, multi-variable models will be validated for reactor control. Phase 2 will culminate with a demonstration of V&V techniques for medium software packages plus a demonstration installed at EBR-II.

Phase 3, FAFTRCS, extends analytical redundancy techniques into the realm of total control.

2.3 The Problems and Proposed Solutions

Four global problems that require resolution for completion of Phase I can be identified. These problems and the proposed solutions are:

- (1) Software reliability - Formal analysis, (validation via testing), and reliability modeling;
- (2) Model certification - Testing in actual environment analysis via dynamic simulation of nuclear plants (DSNP) code;
- (3) Hardware reliability - Ancillary equipment - traditional computer - fault-tolerant properties confirmed via formal analysis and validation testing;
- (4) System validation - Testing in two phases: (1) bench tests to validate design, and (2) in-plant tests to provide acceptance.

Reliability will be analyzed via qualitative and quantitative methods. In the realm of software and, to some extent in hardware, the methods for reliability analysis are developing. Section 3 of this paper presents an overview of the techniques used for this project.

3. DISCUSSION ON FORMAL VERIFICATION

3.1 What are the Goals of Formal Design Verification?

Formal verification is comprised of a rigid mapping between the design specification and its implementation. In the past a typical representation of the implementation was as a set of equations. The verification took the form of a mathematical proof which was difficult to follow. The mapping, via tools such as automated theorem provers, rapidly became convoluted and the technique was suspect. To rectify these problems, a new method for representation is being developed[2,3]. The goals for this representation are:

- (1) MUST map directly to the systems of interest,
- (2) MUST be conducive to formal verification,
- (3) MUST be hierarchical,
- (4) MUST be easy to use.

Goals 1 and 2 make the resulting analysis useable by non-experts in the field of formal analysis by avoiding abstract/convoluted proofs. A hierarchical proof is envisioned as providing a number of benefits, i.e. the reviewer may concentrate on those areas commensurate with personal expertise, many non-critical portions of the system warrant less detailed analysis, and complexity is restrained.

3.2 Integrated H/W & S/W Design Analysis

3.2.1 Discussion

The representation used for analyzing the behavior of the entire systems model will be Petri nets[4]. This representation has been chosen for several reasons. First, the Petri net representation facilitates the hierarchal analysis of an entire system, permitting different levels of abstraction for different parts of the system, as appropriate. Second, the Petri net representation is very general, capable of representing universal Turing machines. In particular, Petri nets have been demonstrated to be powerful enough to represent both synchronous and asynchronous hardware and software systems[2,3,4]. Thus Petri nets can be readily enhanced or restricted to meet the needs of the portion of the entire system being modeled. Third, Petri nets have a graphical representation similar to flow charts and fault-trees, thereby providing a familiar representation which can be readily understood. The formal analysis thus can be examined and validated in a straightforward manner. Fourth, a rather extensive theory for modeling systems has been developed[3], and this theory is used to guide the method of representation and formal analysis. Finally, Petri nets can be used for simulation to help discover properties of the entire system and to assist in the analysis.

Figure 3 is a Petri net representation of the driver routine for inputting sensor data into the FTP. The algorithm is comprised of steps indicated by annotations to the right of each circle. The four circles (places) and bars (transitions) in the upper left portion of the figure represent the hardware involved in the acquisition of data, i.e. the sensor and the bus. Herein, the sensor is assumed to comprise the transducer, wiring, transmitter, and converter. The interesting notion of this representation is that hardware, software and their interaction are equally represented.

Automation of the formal analysis will be effected using an automated theorem program called "Interactive Theorem Prover" which was developed by ANL. Once the Petri net representation was selected, the next task was to define an operational semantic for presentation to ITP. Extensions to Petri net theory as indicated have been proven successful proving various properties of the CSDL design[3].

The above discussion covers the qualitative aspects of the analysis. Quantitative analysis for computer-based systems is equally pervasive. We are currently involved in analyzing various methods for performing software reliability analyses and will continue in this area until a viable approach becomes apparent[6,7].

4. FORMAL VERIFICATION AS APPLIED TO A FAULT-TOLERANT PROCESSOR

4.1 Introduction to the Representation Technique

Petri nets are comprised of places and transitions. As indicated on Figs. 3 & 4, a place is represented by the circle and a transition by a bar. Operationally a place denotes any hardware medium that stores or transports data, i.e. flip-flops, registers, bus. A software place represents any entity which may generate or alter a token, or which generates a control signal. A hardware transition represents a functional module, i.e. an adder, ALU, voter. A software transition is currently conceived as representing modules of code executed in a simplex fashion on individual channels, i.e. conceptually similar to locations in the code where breakpoints would be used during the debugging process. A token is a symbolic expression which can have certain properties associated with it.

Clause templates were created (reference 3, Table 1) for input to the ITP program. The essence of the analysis is that the clausal base is driven by a rule-based manipulator. This manipulator controls flow of tokens through the clause space while maintaining a history of the essential elements of its transversal. In addition it performs various housekeeping functions, i.e. garbage collection and consistency checks.

Table I lists a selection of clauses used to describe the Fig. 3 Petri net to ITP. At present the translation of schematics and wiring information into the Petri representation is manual and as such subject to the frailties of any human intensive effort. Our intent is to develop a tool which would, as a minimum, automate the process of creating clauses from a human developed Petri net. We are currently developing Petri nets on a CAD system and formalizing the representation with automated clause generation as a specification criteria. A future goal, which is as yet unfunded, would be to utilize the CAD output from the CSDL design effort for automated generation of Petri nets.

Table I

Clause Templates for Presentation of a Petri net to the ITP - Interactive Theorem Prover:

```
Transition(xtrans_name,xtype,xcontrol);
Place(xplace_name,xtoken,xtype);
Connection(xtrans_name,xplace_name); or
Connection(xplace_name,xtrans_name);
Function(xtrans_name,xoutput,xcontrol,xfunction);
```

Examples (Figure 3):

```
Place(Sensor_x,Token(Data,Reactor,nil),destructive);
.
.
Place(Comm_x,arbitrary,nondestructive);
Transition(DxSx,Transfer,cntl(Read_sensor_x),Sensor_x);
.
.
Transition(VcxI,Majority_voter,cntl(Vote_x),
      Input_list(Ibus_ox,Input_list(Ibus_xy,Ibus_xz)));
Connection(Dx_Cx,Sensor_x);
.
.
Connection(Comm_x,Vcx1);
```

4.2 An example of Hardware Formal Analysis

The intent of formal analysis is to assure that the system performs according to specification. One method of attaining this assurance is to traverse the annotated paths of the Petri net representation and maintain a history of this traversal. In a system which utilizes four processors, the analysis consists of comparing the histories of each traversal in each processor. Common elements in any two histories indicate a common source for failure which defeats the premise of fault-tolerance.

In the work completed at ANL we have proven various fault-tolerant properties of the CSDL FTP. These proofs make claims about the hardware, i.e. the capability to operate with one fault, if certain assumptions are true, i.e. synchrony, data source

congruence and adequate failure detection. One aspect of using the suggested combination of representation and operation is that all assumptions and assertions are explicitly declared. This accomplishes the goal of facilitating review as the reviewer may focus on determining the acceptability of these declarations.

FROM_OWN More specifically, we have examined the quaded system and have shown that the instruction, if executed by each processor simultaneously, results in (reference 3):

CLAIM:

A majority of processors having congruent data and capable of exhibiting this property in the presence of one fault.

ASSUMPTIONS:

- 1) There exists six fault-containment regions,
- 2) The voted data in a processor will be congruent with respect to the voted data in other processors (provided its voter is fault free),
- 3) Some data from distinct sources that is voted is identical.

4.3 An example of Software Formal Analysis

The software analysis tools lag in development behind those described for the hardware. Figure 4 is a top level Petri net for the operating system (reference 7) used on the FTP. This network has evolved through a number of iterations and will be useful for determination of the following properties:

- 1) Initialization of background tasks in the queue,
- 2) Deterministic completion time for foreground tasks,
- 3) Adequate performance of background tasks, (i.e. a complete set of self tests are performed within a specified time),
- 4) Resume and Activate provide context switching, and
- 5) All tasks involved in a reset, exception or timed interrupt are executed within a predetermined time which is consistent with system response time.

As in the hardware description, the translation of implementation details into the Petri net representation is currently a human intensive effort. ANL is developing a tool which automates the generation of verification conditions for high level code (reference Boyle). The proofs associated with these properties of the top level description of the operating system are comparable to those for the top level of the hardware depicted on the Fig. 3 net. Both sets of proofs are dependent upon supporting proofs in both software and hardware at levels below those shown. Figure 5 is an example of a network which shows the interdependence of the hardware and software at a very detailed level. This figure is included herein for illustration purposes as the representation for this interaction is in a state of flux.

On Fig. 5, the left-hand section of the net represents the hardware and the right-hand section the software. This entire net represents the algorithm associated with the acquisition of data by the individual channels and execution of congruent data exchanges between the acquiring channels. This net is an extension of that in figure 3 in that it represents the interaction between the hardware and software, i.e.

a total system representation. The properties associated with this net are essential to support the claims of fault-tolerance, i.e. that source congruence of data is preserved.

The convention of using dashed lines to represent the flow of control between the hardware and software networks results from the historical distinction between hardware and software analysis. This distinction is currently being debated as the lines of distinction in our analysis are becoming less defined.

5. CONCLUSION

The intent of this paper was twofold: 1) to introduce one approach to determining total system reliability and 2) to report some results for a novel approach to formal analysis of a system. The total systems approach to reliability demands the coordination of many diverse disciplines, e.g. fault-tolerant system design, computer science, formal analysis (both software and hardware), reliability analysis (both software and hardware). The project described in section 2 is largely funded by DOE with supplemental funding from both NRC and EPRI and is intended to provide a demonstration of a total systems approach to reliability. The inclusion of a discussion on Petri net representation was intended to demonstrate the power of the annotated Petri net. Specifically, the capability for representation of the interaction between hardware and software, provides the vehicle for an extension of formal analytical techniques to apply to total system analysis.

6. ACKNOWLEDGEMENTS

This report encompasses the combined efforts of those involved in the FAFTRCS project at ANL and CSDL. Specifically the efforts by J. Kljaich, A.S. Wojcik and B.T. Smith must be mentioned as they are responsible for selection of the Petri net as the tool for modeling as well as development of the formalism for annotating these representations.

7. REFERENCES

1. Smith, T. B., "Fault-Tolerance Processor Concepts and Operation," CSDL-P-1727 (1983).
2. Wojcik, A. S., Kljaich, J., and Srinivas, N. (1983), "A Formal Design Verification System Based on an Automated Reasoning System," Proceedings of the 20th Design Automation Conf.
3. Kljaich, J., "Doctoral Thesis at Illinois Institute of Technology," to be published.
4. Peterson, J. L. (1981) Petri Net Theory and the Modeling of Systems, Prentice-Hall, Inc.
5. Alger, L. S., Lala, J. H., "A Real Time Operating System for a Nuclear Power Plant Computer," Proceedings of EPRI Seminar: Power Plant Digital Control and Fault-Tolerant Microcomputers (1985).
6. Butler, R. W. "The Semi-Markov Unreliability Range Evaluator (SURE) Program," NASA Tech. Mem 86261 (1984).
7. Farr, W. H., "A Survey of Software Reliability Modeling and Estimation" NSWC-TR-82-171 (1983)".

FAFTRCS CONCEPTUAL DESIGN
OVERVIEW OF CSDL FTP

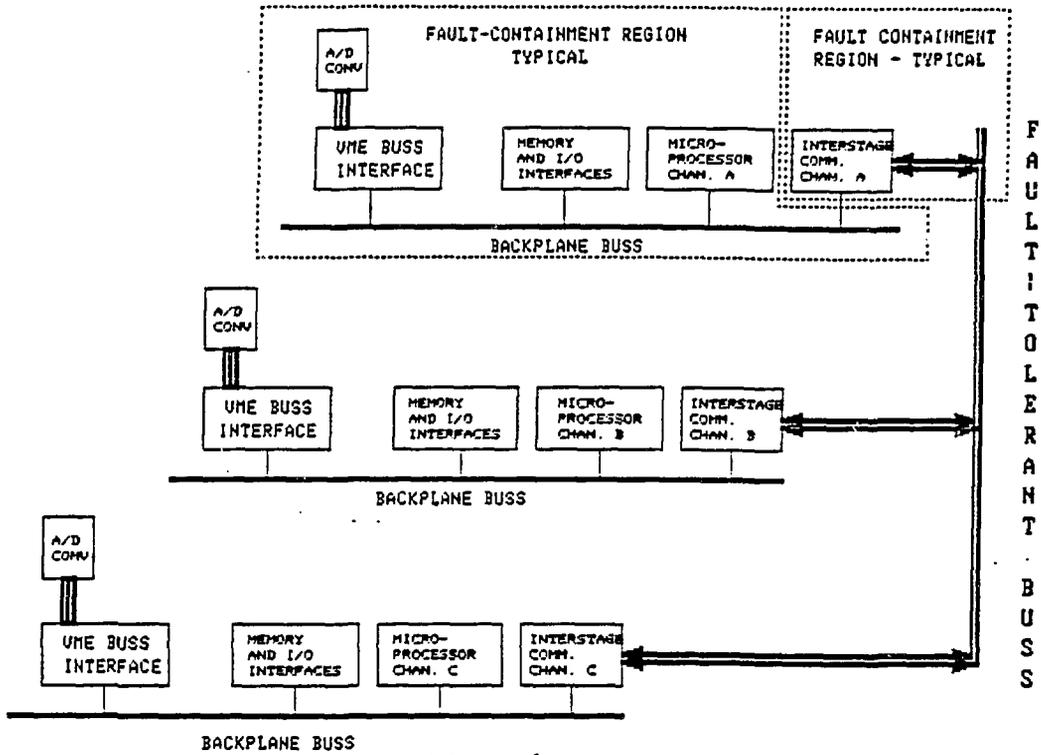


Figure 1

FAFTRCS CONCEPTUAL DESIGN - FT² SYSTEM OVERVIEW

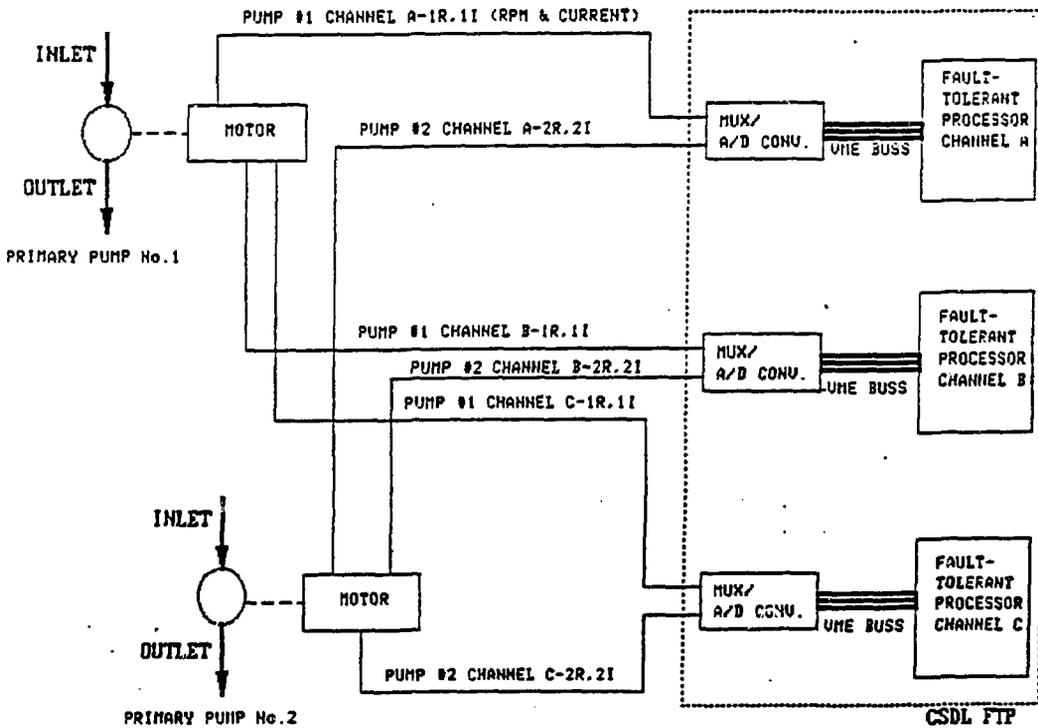


Figure 2