NASA Langley's Research and Technology-Transfer Program in Formal Methods

Ricky W. Butler
James L. Caldwell
Victor A. Carreño
C. Michael Holloway
Paul S. Miner
Assessment Technology Branch
NASA Langley Research Center
Hampton, Virginia

Ben L. Di Vito
ViGYAN Inc.
Hampton, Virginia

Abstract

This paper presents an overview of NASA Langley's research program in formal methods. The major goals of this work are to make formal methods practical for use on life critical systems, and to orchestrate the transfer of this technology to U.S. industry through use of carefully designed demonstration projects. Several direct technology transfer efforts have been initiated that apply formal methods to critical subsystems of real aerospace computer systems. The research team consists of five NASA civil servants and contractors from Odyssey Research Associates, SRI International, and ViGYAN Inc.

1 Rationale For Formal Methods Research Program

NASA Langley Research Center has been developing techniques for the design and validation of flight critical systems for over two decades. Although much progress has been made in developing methods to accommodate physical failures, design flaws remain a serious problem [1, 2, 3, 4, 5, 6, 7]. A 1991 report by the National Center For Advanced Technologies[1] identified "Provably Correct System Specification" and "Verification Formalism For Error-Free Specification" as key areas of research for future avionics software and ultrareliable electronics systems [8].

[1] A technical council funded by the Aerospace Industries Association of America (AIA) that represents the major U.S. aerospace companies engaged in the research, development and manufacture of aircraft, missiles and space systems, and related propulsion, guidance, control and other equipment.

1.1 Why Formal Methods Are Necessary

Digital systems (both hardware and software) are notorious for their unpredictable and unreliable behavior:

> Studies have shown that for every six new large-scale software systems that are put into operation, two others are cancelled. The average software development project overshoots its schedule by half; larger projects generally do worse. And three quarters of all large systems are "operating failures" that either do not function as intended or are not used at all.
>
> Despite 50 years of progress, the software industry remains years–perhaps decades–short of the mature engineering discipline needed to meet the demands of an information-age society[6].

Lauren Ruth Wiener describes the software problem in her book, *Digital Woes: Why We Should Not Depend Upon Software*:

> Software products—even programs of modest size—are among the most complex arti-

> facts that humans produce, and software development projects are among our most complex undertakings. They soak up however much time or money, however many people we throw at them.
>
> The results are only modestly reliable. Even after the most thorough and rigorous testing some bugs remain. We can never test all threads through the system with all possible inputs[5].

The hardware industry also faces serious difficulties, as evidenced by the recent design error in the Pentium floating point unit. In response to an outcry over the design flaw in the Pentium floating point unit, Intel's President, Andy Grove, wrote on the comp.sys.intel Internet bulletin board:

> After almost 25 years in the microprocessor business, I have come to the conclusion that no microprocessor is ever perfect; they just come closer to perfection with each stepping. In the life of a typical microprocessor, we go thru [sic] half a dozen or more such steppings....

In a recent Washington Post article, Michael Schrage wrote:

> Pentium type problems will prove to be the rule—rather than the isolated, aberrant exceptions—as new generations of complex hardware and software hit the market. More insidious errors and harmful bugs are inevitable. That is the new reality[9].

For life critical systems, errors may mean disaster. The potential for errors is high, because these systems must not only perform their functions correctly, but also must be able to recover from the effects of failing components. Often the physical fault tolerance features of these systems are more complex and susceptible to design errors than any of the basic functions of the system. John Rushby writes:

> Organization of redundancy and fault-tolerance for ultra-high reliability is a challenging problem: redundancy management can account for half the software in a flight control system and, if less than perfect can itself become the primary source of system failure [10].

In a comprehensive assessment of formal methods [11], John Rushby discusses several notorious examples of such failures. These include the following:

- The asynchronous operation of the AFTI-F16 and sensor noise led each channel to declare the other channels failed in flight test 44. The plane was flown home on a single channel. Other potentially disastrous bugs were detected in flight tests 15 and 36.
- The HiMAT crash landed without its landing gear due to a design flaw. The problem was traced to a timing change in the software that had survived extensive testing.
- A bug in the YC-14 redundancy management was found during flight test. The bug caused a large mistracking between redundant channels.
- In flight tests of the X31, the control system went into a reversionary mode four times in the first nine flights, usually due to a disagreement between the two air data sources.
- The nationwide saturation of the AT&T switching systems on January 15, 1990 was caused by a timing problem in a fault-recovery mechanism.
- The first Shuttle mission (STS-1) was scrubbed because the fifth backup computer could not be synchronized with the other four.

Three basic strategies are advocated for handling design flaws in life critical systems:

1. Testing (Lots of it)
2. Design Diversity (i.e. software fault tolerance: N-version programming, recovery blocks, etc.)
3. Fault Avoidance (i.e. formal specification and verification, automatic program synthesis, reusable modules)

The problem with life testing is that in order to measure ultrareliability one must test for exorbitant amounts of time. For example, to measure a 10^{-9} probability of failure for a 1 hour mission one must test for more than 10^9 hours (114,000 years).

The basic idea behind design diversity is to use separate design and implementation teams to produce multiple versions from the same specification. At run-time, non-exact threshold voters are used to mask the effect of a design error in one of the versions. The hope is that the design flaws will manifest errors independently or nearly so. By assuming independence, one can obtain ultrareliable-level estimates of system reliability, even with failure rates for the individual versions on the order of 10^{-4}/hour. Unfortunately, the independence assumption has been rejected at the 99% confidence level in several experiments for low reliability software [12, 13].

Furthermore, the independence assumption cannot be validated for high reliability software because of the exorbitant test times required. If one cannot assume independence then one must measure correlations. This is infeasible as well; it requires as much testing time as life-testing the system, because the correlations must be in the ultrareliable region in order for the system to be ultrareliable. Therefore, it is not possible, within feasible amounts of testing time, to establish that design diversity achieves ultrareliability. Consequently, design diversity can create an "illusion" of ultrareliability without actually providing it. For a more detailed discussion, see [14].

We believe that formal methods offer the only intellectually defensible method for handling design faults. Since the often quoted $1 - 10^{-9}$ reliability is clearly beyond the range of quantification, we have no choice but to develop life critical systems in the most rigorous manner available to us, which is use of formal methods.

1.2 What are Formal Methods

Engineering relies heavily on mathematical models and calculation to make judgments about designs. This is in stark contrast to the way in which software systems are designed—with *ad hoc* technique and after-implementation testing. Formal methods bring to software and hardware design the same advantages that other engineering endeavors have exploited: mathematical analysis based on models. Formal methods are used to specify and model the behavior of a system and to formally verify that the system design and implementation satisfy functional and safety properties. In principle, these techniques can produce error-free design; however, this requires a complete verification from the requirements down to the implementation, which is rarely done in practice.

Thus, formal methods are the applied mathematics of computer systems engineering. They serve a similar role in computer design as Computational Fluid Dynamics (CFD) plays in aeronautical design, providing a means of calculating and hence predicting what the behavior of a digital system will be prior to its implementation.

The tremendous scientific potential of formal methods has been recognized by theoreticians for a long time, but the formal techniques have remained the province of a few academicians, with only a few exceptions such as the Transputer [15] and the IBM CICS project [16]. The first five years of NASA Langley's program have advanced the capabilities of formal methods to the point where commercial exploitation is near.

There are many different types of formal methods with various degrees of rigor. The following is a useful (first-order) taxonomy of the degrees of rigor in formal methods:

- Level-1: Formal specification of all or part of the system.
- Level-2: Formal specification at two or more levels of abstraction and paper and pencil proofs that the detailed specification implies the more abstract specification.
- Level-3: Formal proofs checked by a mechanical theorem prover.

Level 1 represents the use of mathematical logic, or a specification language that has a formal semantics, to specify the system. This can be done at several levels of abstraction. For example, one level might enumerate the required abstract properties of the system, while another level describes an implementation that is algorithmic in style.

Level 2 formal methods go beyond Level 1 by developing pencil-and-paper proofs that the concrete levels logically imply the abstract, property-oriented levels. *Level 3* is the most rigorous application of formal methods. Here one uses a semi-automatic theorem prover to make sure that all of the proofs are valid. The Level 3 process of *convincing* a mechanical prover is really a process of developing an argument for an ultimate skeptic who must be shown every detail.

It is important to realize that formal methods are not an all-or-nothing approach. The application of formal methods to the most critical portions of a system is a pragmatic and useful strategy. Although a complete formal verification of a large complex system is impractical at this time, a great increase in confidence in the system can be obtained by the use of formal methods at key locations in the system. For more information on the basic principles of formal methods, see [17].

2 Goals of Our Program, Strategy, and Research Team

The major goals of the NASA Langley research program are to make formal methods practical for use on life critical systems developed in the United States, and to orchestrate the transfer of this technology to industry through use of carefully designed demonstration projects. Our intention is to concentrate our research efforts on the technically challenging areas of digital flight-control systems design that are currently beyond the state-of-the-art, while initiating demonstration projects in problem domains where current formal methods are adequate. The challenge of the demonstration projects should not be underestimated. That which is feasible for experts that have developed the tools and methods is often difficult for practitioners in the aerospace industry. There is often a long "learning curve" associated with the tools, the tools are not production-quality, and the tools have few or no examples for specific problem domains. Therefore, we are setting up cooperative efforts between industry and the developers of the formal methods to facilitate the technology transfer process.

This strategy leverages the huge investment of ARPA and the National Security Agency in development of tools and concentrates on the problems specific to the aerospace problem domain. NASA Langley has not sponsored the development of any general-purpose theorem provers. However, the technology transfer projects have lead to significant improvements in the Prototype Verification System (PVS) theorem prover[10] that SRI International (SRI) is developing. Several domain-specific tools are being sponsored: (1) Tablewise, (2) VHDL-analysis tool, and (3) DRS. These tools are discussed in later sections.

It is also important to realize that formal methods include a large class of mathematical techniques and tools. Methods appropriate for one problem domain may be totally inappropriate for other problem domains. The following are some of the specific domains in which our program has concentrated: (1) architectural-level fault tolerance, (2) clock-synchronization, (3) interactive consistency, (4) design of hardware devices such as microprocessors, memory management units, DMA controllers, (5) asynchronous communication protocols, (6) design and verification of application-specific integrated circuits (ASICS), (7) Space Shuttle software, (8) navigation software, (9) decision tables, (10) railroad signaling systems.

We are also interested in applying formal methods to many different portions of the life-cycle, such as (1) requirements analysis, (2) high-level design, (3) detailed design, and (4) implementation.

Often, there is a sizable effort associated with the development of the background mathematical theories needed for a particular problem domain. Although such theories are reusable and in the long run can become "cost-effective", the initial costs can be a deterrent for industry. Therefore, one of the goals of the NASA Langley program is to build a large body of background theories needed for aerospace applications.

We also have been involved with standards activities in order to strengthen the United States commitment to safety.

2.1 Technology Transfer

The key to successful technology transfer is building a cooperative partnership with a customer. In order for this partnership to work, NASA Langley must become directly involved in specific problem domains of the aerospace industry[2]. NASA must also effectively communicate its basic research accomplishments in a manner that reveals a significant potential benefit to the aerospace community. Equally important is the need for industry to make an investment to work together with NASA on joint projects to devise demonstration projects that are realistic and practical. The ultimate goal of our technology transfer process is for formal methods to become the "state-of-the-practice" for U.S. industry development of ultrareliable digital avionics systems. However, before we can develop new tools and techniques suitable for adoption by industry, we must work with the system developers in industry to understand their needs. We must also overcome the natural skepticism that industry has of any new technology.

Our basic approach to technology transfer is as follows. The first step is to find an industry representative who has become interested in formal methods, believes that there is a potential benefit of such methods, and is willing to work with us. The next step is to fund our formal methods research team to apply formal methods to an appropriate example application. This process allows the industry representative to see what formal methods are and what it has to offer, and it allows us (the formal methods team) to learn the design and implementation details of state-of-the-practice components so we can better tailor our tools and techniques to industry's needs. If the demonstration project reveals a significant potential benefit, the next stage of the technology transfer process is for the industry representative to initiate an internal formal methods program, and begin a true cooperative partnership with us.

Another important part of our technology transfer strategy is working with the Federal Aviation Ad-

[2]To date, our efforts have concentrated on the aerospace industry, but we are actively seeking partners from other industries also.

ministration (FAA) to update certification technology with respect to formal methods. If the certification process can be redefined in a manner that awards credit for the use of formal methods, a significant step towards the transfer of this technology to the commercial aircraft industry will have been accomplished.

Langley has also been sponsoring a series of workshops on formal methods. The first workshop, held in August 1990, focused on building cooperation and communication between U.S. formal methods researchers[18]. The second, held in August 1992, focused on education of the U.S. aerospace industry about formal methods[19]. A third workshop will be held in May 1995.

Another component of our technology transfer strategy, is to use the NASA's Small Business Innovative Research (SBIR) program to assist small businesses in the development of commercially viable formal methods tools and techniques. The first contracts under the program began in early 1994.

Finally, to facilitate technology transfer, information on NASA Langley's formal methods research is available on the Internet via either anonymous FTP or World Wide Web. PostScript and DVI versions of many research papers are available through anonymous FTP on machine `deduction.larc.nasa.gov` (IP address: `128.155.18.16`) in directory `pub/fm`. This directory, and much more information, is also available through World Wide Web, using the following Uniform Resource Locator:

`http://atb-www.larc.nasa.gov/fm-top.html`

2.2 FAA/RTCA Involvement

As the federal agency responsible for certification of civil air transports, the FAA shares our interest in promising approaches to engineering and validating ultrareliable flight-control systems. Additionally, because the FAA must approve any new methodologies for developing life critical digital systems for civil air transports, their acceptance of formal methods is a necessary precursor to its adoption by industry system designers. We are working with Pete Saraceni of the FAA Technical Center and Mike DeWalt, FAA National Resource Specialist for Software, to insure that our program is relevant to the certification process. The FAA has co-sponsored some of our work. John Rushby of SRI gave a tutorial on formal methods at an FAA Software Advisory Team (SWAT) meeting at their request. The SWAT team suggested that we include an assessment of formal methods in an ongoing Guidance Control Software (GCS) experiment in our branch; Odyssey Research Associates (ORA) developed a formal specification of the GCS application.

John Rushby has written a chapter for the FAA Digital Systems Validation Handbook Volume III on formal methods[20]. The handbook provides detailed information about digital system design and validation and is used by the FAA certifiers. In preparation for this chapter, Rushby produced a comprehensive analysis of formal methods [11].

George Finelli, the former assistant Branch Head of the System Validation Methods Branch (the Branch in which the formal methods team worked before NASA Langley's reorganization in 1994) and a member of the RTCA committee formed to develop DO–178B, together with Ben Di Vito (VíGYAN Inc.), was instrumental in including formal methods as an alternate means of compliance in the DO–178B standard.

Currently, members of the Langley staff are involved in RTCA committees SC-180 (Airborne Electronic Hardware) and SC-182 (Minimal Operating Performance Standard for an Airborne Computer Resource).

2.3 Team

The Langley formal methods program involves both local researchers and industrial / academic researchers working under contract to NASA Langley. Currently the local team consists of five civil servants and one contractor (VíGYAN Inc.). The lead NASA Langley formal methods researcher, Ricky W. Butler, may be contacted through electronic mail to `R.W.Butler@LaRC.NASA.GOV`.

NASA Langley has recently awarded two five-year task-assignment contracts specifically devoted to formal methods (from the competitive NASA RFP 1-132-DIC.1021). The selected contractors were SRI International (SRI) and Odyssey Research Associates (ORA). This was a follow-on contract from the previous competitive contract that had awarded three contracts to SRI, ORA, and Computational Logic Inc. (CLI).

3 Current Technology Development and Transfer Projects

3.1 AAMP5/AAMP-FV Project

In 1993, NASA Langley initiated a joint project involving Collins Commercial Avionics and SRI International. The goal was to investigate the application of formal techniques to a commercial microprocessor design, the Collins AAMP5 microprocessor. The AAMP5 is the latest member of the CAPS/AAMP family of microprocessors and is object code compatible with the AAMP2 processor [21]. The CAPS/AAMP family of microprocessors has been widely used by the commercial and military aerospace industries. Some examples of use of earlier members of the family include: (1) Boeing 747-400 Integrated Display System (IDS), (2) Boeing 737-300 Electronic Flight Instrumentation System (EFIS), (3) Boeing 777 Flight Control Backdrive, (4) Boeing 757,767 Autopilot Flight Director System (AFDS), and (5) military and commercial Global Positioning (GPS) Systems. The first phase of the project consisted of the formal specification of the AAMP5 instruction set and microarchitecture using SRI's PVS [22, 23] While formally specifying the microprocessor, two design errors were discovered in the microcode. These errors were uncovered as a result of questions raised by the formal methods researchers at Collins and SRI while seeking to formally specify the behavior of the microprocessor[24]. The Collins formal methods team believes that this effort has prevented two significant errors from going into the first fabrication of the microprocessor.

The second phase of the project consisted of formally verifying the microcode of a representative subset of the AAMP5 instructions. Collins seeded two errors in the microcode provided to SRI in an attempt to assess the effectiveness of formal verification. Both of these errors (and suggested corrections) were discovered while proving the microcode correct[24]. It is noteworthy that both the level 2 and level 3 applications of formal methods were successful in finding bugs. Based on the success of the AAMP5 project, a new effort has been initiated with Rockwell-Collins to apply formal methods in the design level verification of a microprocessor, currently designated as AAMP-FV.

3.2 Tablewise Project

Under NASA funding, Odyssey Research Associates is working with Honeywell Air Transport Systems Division (Phoenix) to study the incorporation of formal methods into the company's software development processes. Because Honeywell uses decision tables to specify the requirements and designs for much of their software[3], ORA is developing a prototype tool, called Tablewise, to analyze the characteristics of decision tables. Tablewise uses a generalization of Binary Decision Diagrams to determine if a particular table is exclusive (for every combination of parameter values, at most one action can be chosen) and exhaustive (for every combination of parameter values, at least one action can be chosen). The tool is also capable of automatically generating documentation and Ada code from a decision table. We consider this a level 3 application of formal methods: although a general purpose prover is not used, the analysis is mechanized in a computer program.

In 1995, ORA will develop algorithms to handle advanced analysis of decision tables. Two particular areas of analysis that will be considered are testing of additional properties of tables and techniques for efficiently handling partitioned tables. The Honeywell personnel involved in the project hope that the concepts developed in the Tablewise project can be incorporated into an industrial-strength tool that will significantly reduce the effort required to develop new software.

3.3 Union Switch and Signal

As part of a joint research agreement, NASA Langley formal methods researchers are collaborating with engineers at Union Switch and Signal (US&S) to use formal methods in the design of railway switching and control applications. Railway switching control systems, like digital flight control systems, are safety critical systems. US&S is the leading U.S. supplier of railway switching control systems. Their Advanced Technology Group, lead by Dr. Joseph Profeta, has applied formal methods in past efforts and turned to NASA for expertise in integrating these techniques into their next generation products.

The initial project, started in 1993, was a cooperative effort between NASA, US&S, and Odyssey Research Associates. The result of this first year's work was a formal mathematical model of a railway switching network, defined in two levels. The top level of the model provides the mechanisms for defining the basic concepts: track, switches, trains and their positions and control liners of a train (i.e. how far down the track it has clearance to travel.) The second level is a formalization of the standard scheme used in railroad control, the block model control system. A level 2 proof that the fixed block control system is "safe" with respect to the top level model has been completed. Models of US&S proprietary control schemes were also formulated.

The European formal methods community has addressed safety properties of certain components of railroad control systems, but the work there has typically

[3] A decision table is a tabular format for defining the rules that choose a particular action to perform based on the values of certain parameters.

been at lower levels. The cooperative work with US&S is unique in that a high level model of a railroad system has been described and used to analyze the safety of various control schemes.

The next phase of the collaborative effort will concentrate on formal modeling and analysis of the fault-tolerant core of US&S's next generation fail-stop control architecture.

3.4 Space Applications

A team spread across three NASA centers has been formed to study the application and technology transfer of formal methods to NASA space programs. A consortium of researchers and practitioners from LaRC, JSC, and JPL, together with support from Loral Space Information Systems, SRI International, and VíGYAN Inc., has been actively pursuing this objective since late 1992. The near term goal is to define and carry out pilot projects using portions of existing large-scale space programs. The long term goal is to enable organizations such as Loral to reduce formal methods to practice on programs of national importance.

The NASA Formal Methods Demonstration Project for Space Applications focuses on the use of formal methods for requirements analysis because the team believes that formal methods are more practically applied to requirements analysis than to late-lifecycle development phases [25]. A series of trial projects was conducted and cost effectiveness data were collected. The team's efforts in 1993 were concentrated on a single pilot project (discussed in a subsequent section), while efforts beginning in 1994 have been more diffuse.

NASA Langley's primary role in 1994 included support for two Space Shuttle software change requests (CR). One CR concerns the integration of new Global Positioning System (GPS) functions while the other concerns a new function to control contingency aborts known as Three Engine Out (3 E/O). Both of these tasks involve close cooperation among formal methods researchers at NASA Langley, VíGYAN Inc., and SRI International with requirements analysts from Loral Space Information Systems.

The Space Shuttle is to be retrofitted with GPS receivers in anticipation of the TACAN navigation system being phased out by the DoD. Additional navigation software will be incorporated to process the position and velocity vectors generated by these receivers. A decision was made to focus the trial formal methods task on just a few key areas because the CR itself is very large and complex. A set of preliminary formal specifications was developed for the new Shuttle navigation principal functions known as GPS Receiver State Processing and GPS Reference State Processing, using the language of SRI's Prototype Verification System (PVS). While writing the formal specifications, 43 minor discrepancies were detected in the CR and these have been reported to Loral requirements analysts.

The Three Engine Out (3 E/O) Task is executed each cycle during powered flight until either a contingency abort maneuver is required or progress along the powered flight trajectory is sufficient to preclude a contingency abort even if three main engines fail. The 3 E/O task consists of two parts: 3 E/O Region Selection and 3 E/O Guidance. 3 E/O Region Selection is responsible for selecting the type of external tank (ET) separation maneuver and assigning the corresponding region index. 3 E/O guidance monitors ascent parameters and determines if an abort maneuver is necessary.

We have developed and analyzed a formal model of the series of sequential maneuvers that comprise the 3 E/O algorithm. To date, 20 potential issues have been found, including undocumented assumptions, logical errors, and inconsistent and imprecise terminology. These findings are listed as potential issues pending review by the 3 E/O requirements analyst.

The GPS and 3 E/O tasks have continued into 1995. We hope to get formal methods incorporated as a requirements analysis technique for Space Shuttle software. In addition, NASA Langley contributed to a NASA guidebook under development by the intercenter team. The first volume of the guidebook is intended for managers of NASA projects who will be using formal methods in requirements analysis activities. A second volume is planned that will be aimed at practitioners. NASA will publish the first volume early in 1995, with the second volume expected by early 1996.

3.5 NASA Small Business Innovative Research Program

In 1993, a formal methods subtopic was a part of the NASA Small Business Innovative Research (SBIR) solicitation. Two proposals were selected for 6-month Phase I funding for 1994: *VHDL Lightweight Tools*, by Odyssey Research Associates, and *DRS — Derivation Reasoning System, A Digital Design Derivation System for Hardware Synthesis*, by Derivation Systems, Inc. of Bloomington, Indiana. After the completion of the Phase I efforts, both companies were selected for continued Phase II funding. Contracts for these efforts just recently began.

4 Past Efforts

This section describes previous work in each of the following four focus areas: fault-tolerant systems, verification of software, verification of hardware devices, and civil air transport requirements specification. This section omits much of the early work described at COMPASS 91 [26].

4.1 Fault-tolerant Systems

The goal of this focus area was to create a formalized theory of fault tolerance including redundancy management, clock synchronization, Byzantine agreement, voting, etc. Much of the theory developed here is applicable to future fault-tolerant systems designs. A detailed design of a fault-tolerant reliable computing base, the Reliable Computing Platform (RCP), has been developed and proven correct. It is hoped that the RCP will serve as a demonstration of the formal methods process and provide a foundation that can be expanded and used for future aerospace applications.

The RCP architecture was designed in accordance with a system design philosophy called "Design For Validation" [27, 28]. The basic tenets of this design philosophy are as follows:

1. A system is designed in such a manner that complete and accurate models can be constructed to estimate critical properties such as reliability and performance. All parameters of the model that cannot be deduced from the logical design must be measured. All such parameters must be measurable within a feasible amount of time.

2. The design process makes tradeoffs in favor of designs that minimize the number of parameters that must be measured in order to reduce the validation cost. A design that has exceptional performance properties yet requires the measurement of hundreds of parameters (for example, by time-consuming fault-injection experiments) would be rejected over a less capable system that requires minimal experimentation.

3. The system is designed and verified using rigorous mathematical techniques. It is assumed that the formal verification makes system failure due to design faults negligible so the reliability model does not include transitions representing design errors.

4. The reliability (or performance) model is shown to be accurate with respect to the system implementation. This is accomplished analytically not experimentally.

Thus, a major objective of this approach is to minimize the amount of experimental testing required and maximize the ability to reason mathematically about correctness of the design. Although testing cannot be eliminated from the design/validation process, *the primary basis of belief in the dependability of the system must come from analysis rather than from testing.*

4.1.1 The Reliable Computing Platform

The Reliable Computing Platform dispatches control-law application tasks and executes them on redundant processors. The intended applications are safety critical with reliability requirements on the order of $1 - 10^{-9}$. The reliable computing platform performs the necessary fault-tolerant functions and provides an interface to the network of sensors and actuators.

The RCP operating system provides the applications software developer with a reliable mechanism for dispatching periodic tasks on a fault-tolerant computing base that *appears* to him as a single ultrareliable processor. The top level of the hierarchy describes the operating system as a function that sequentially invokes application tasks. This view of the operating system will be referred to as the *uniprocessor specification (US)*, which is formalized as a state transition system and forms the basis of the specification for the RCP. Fault tolerance is achieved by voting results computed by the replicated processors operating on the same inputs. Interactive consistency checks on sensor inputs and voting of actuator outputs require synchronization of the replicated processors. The second level in the hierarchy (RS) describes the operating system as a synchronous system where each replicated processor executes the same application tasks. The existence of a global time base, an interactive consistency mechanism and a reliable voting mechanism are assumed at this level.

Level 3 of the hierarchy (DS) breaks a frame into four sequential phases. This allows a more explicit modeling of interprocessor communication and the time phasing of computation, communication, and voting. At the fourth level (DA), the assumptions of the synchronous model are discharged through use of the interactive-convergence clock synchronization algorithm [29].

In the LE model, a more detailed specification of the activities on a local processor are presented. In particular, three areas of activity are elaborated in detail: (1) task dispatching and execution, (2) minimal voting, and (3) interprocessor communication via mailboxes. An intermediate model, DA_minv, that simplified the construction of the LE model was used.

Of primary importance in the LE specification is the use of a memory management unit by the local executive in order to prevent the overwriting of incorrect memory locations while recovering from the effects of a transient fault.

The top two levels of the RCP were originally formally specified in standard mathematical notation and connected via mathematical (i.e. level 2 formal methods) proof [30, 31, 32]. Under the assumption that a majority of processors is working in each frame, the proof establishes that the replicated system computes the same results as a single processor system not subject to failures. Sufficient conditions were developed that guarantee that the replicated system recovers from transient faults within a bounded amount of time. SRI subsequently generalized the models and constructed a mechanical proof in EHDM [33]. Next, the local team developed the third and fourth level models. The top two levels and the two new models (i.e. DS and DA) were then specified in EHDM and all of the proofs were done mechanically using the EHDM 5.2 prover [34, 35].

Both the DA_minv model and the LE model were specified formally and have been verified using the EHDM verification system[36]. All RCP specifications and proofs are available electronically via the Internet using anonymous FTP or World Wide Web (WWW) access. Anonymous FTP access is available through the host `deduction.larc.nasa.gov` using the path `pub/fm/larc/RCP-specs`. WWW access to the FTP directory is provided through the NASA Langley Formal Methods Program home page: `http://atb-www.larc.nasa.gov/fm-top.html`

4.1.2 Clock Synchronization

The redundancy management strategies of virtually all fault-tolerant systems depend on some form of voting, which in turn depends on synchronization. Although in many systems the clock synchronization function has not been decoupled from the applications (e.g. the redundant versions of the applications synchronize by messages), research and experience have led us to believe that solving the synchronization problem independently from the applications design can provide significant simplification of the system [37, 38]. The operating system is built on top of this clock-synchronization foundation. Of course, the correctness of this foundation is essential. Thus, the clock synchronization algorithm and its implementation are prime candidates for formal methods. The verification strategy shown in figure 1 is being explored.

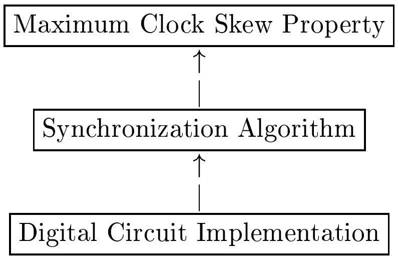

Figure 1: Hierarchical Verification of Clock Synchronization

The top-level in the hierarchy is an abstract property of the form:

$$\forall \text{ non-faulty } p, q : |C_p(t) - C_q(t)| < \delta$$

where δ is the maximum clock skew guaranteed by the algorithm as long as a sufficient number of clocks (and the processors they are attached to) are working. The function $C_p(t)$ gives the value of clock p at real time t. The middle level in the hierarchy is a mathematical definition of the synchronization algorithm. The bottom level is a detailed digital design of a circuit that implements the algorithm. The bottom level is sufficiently detailed to make translation into silicon straight forward.

The verification process involves two important steps: (1) verification that the algorithm satisfies the maximum skew property and (2) verification that the digital circuitry correctly implements the algorithm. The first step was completed by SRI International. The first such proof was accomplished during the design and verification of SIFT [29]. The proof was done by hand in the style of journal proofs. More recently this proof step was mechanically verified using the EHDM theorem prover[39, 40]. In addition, SRI mechanically verified Schneider's clock synchronization paradigm [41] using EHDM[42, 43]. A further generalization was found at NASA Langley [44][4]. The design of a digital circuit to distribute clock values in support of fault-tolerant synchronization was completed by SRI and was partially verified.[5] CLI reproduced the SRI verification of the interactive convergence algorithm using the Boyer-Moore theorem prover [45].

NASA Langley researchers designed and implemented a fault-tolerant clock synchronization circuit capable of recovery from transient faults [46, 47, 44]. The top-level specification for the design is the EHDM

[4] The bounded delay assumption was shown to follow from the other assumptions of the theory.

[5] Unlike the NASA circuit, the SRI intent is that the convergence algorithm be implemented in software.

verification of Schneider's paradigm. The circuit was implemented with programmable logic devices (PLDs) and FOXI fiber optic communications chips [48].

Using a combination of formal techniques, a verified clock synchronization circuit design has also been developed[49]. The principal design tool was the Digital Design Derivation system (DDD) developed by Indiana University[50]. Some design optimizations that were not possible within DDD were verified using PVS.

4.1.3 Byzantine Agreement Algorithms

Fault-tolerant systems, although internally redundant, must deal with single-source information from the external world. For example, a flight control system is built around the notion of feedback from physical sensors such as accelerometers, position sensors, and pressure sensors. Although these can be replicated (and they usually are), the replicates do not produce identical results. To use bit-by-bit majority voting, all of the computational replicates must operate on identical input data. Thus, the sensor values (the complete redundant suite) must be distributed to each processor in a manner which guarantees that all working processors receive exactly the same value even in the presence of some faulty processors. This is the classic Byzantine Generals problem [51]; algorithms to solve the problem are called Byzantine agreement algorithms. CLI investigated the formal verification and implementation of such algorithms. They formally verified the original Marshall, Shostak, and Lamport version of this algorithm using the Boyer Moore theorem prover [52]. They also implemented this algorithm down to the register-transfer level and demonstrated that it implements the mathematical algorithm [53], and then subsequently verified the design down to a hardware description language HDL developed at CLI [54]. A more efficient mechanical proof of the oral messages algorithm was also developed by SRI[55].

ORA also investigated the formal verification of Byzantine Generals algorithms. They focused on the practical implementation of a Byzantine-resilient communications mechanism between Mini-Cayuga microprocessors [56, 57]. The Mini-Cayuga is a small but formally verified microprocessor developed by ORA. It is a research prototype and has not been fabricated. The communications circuitry would serve as a foundation for a fault-tolerant architecture. It was designed assuming that the underlying processors were synchronized (say by a clock synchronization circuit). The issues involved with connecting the Byzantine communications circuit with a clock synchronization circuit and verifying the combination has not yet been explored.

Thambidurai and Park [58] introduced a fault model that classified faults into three categories: asymmetric, symmetric, and benign. They further suggested the need for and developed an algorithm that had capabilities beyond that of the earlier Byzantine generals algorithms. In particular, their algorithm can mask the effects of a less severe class of faults, in a more effective way. SRI has formally verified an improved version of this algorithm [59, 60, 61]

The newly developed hybrid-fault theory was then applied to the analysis of the Charles Stark Draper Labs "Fault-Tolerant Processor" (FTP). A unique feature of this architecture is its use of "interstages" to relay messages between processors. These are significantly smaller than a processor and lead to an asymmetric architecture that is far more efficient than the traditional Byzantine agreement architectures. The SRI work not only formalized the existing informal analysis but extended it to cover a wider range of faulty behavior[62].

Also SRI subsequently generalized their clock synchronization work to encompass the hybrid fault model [63].

4.2 Verification of Software

Our past software verification projects are described in this section.

4.2.1 Formal Specification of Space Shuttle Jet Select

NASA Langley worked with NASA Johnson Space Center and the Jet Propulsion Laboratory (JPL) in a study to explore the feasibility and utility of applying mechanically-supported formal methods to requirements analysis for space applications. The team worked jointly to develop a formal specification of the Jet Select function of the NASA Space Shuttle, which is a portion of the Shuttle's Orbit Digital Auto-Pilot (DAP). Although few proofs were produced for this specification, 46 issues were identified and several minor errors were found in the requirements. A second specification was produced for an abstract (i.e., high level) representation of the Jet Select requirements. This abstraction, along with the 24 proofs of key properties, was accomplished in under 2 work months, and although it only uncovered 6 issues, several of these issues were significant. Even this level 1 application of formal methods was able to uncover hidden problems in a highly critical and mature FSSR specification for Shuttle.

This project impressed several key members of the Shuttle software community that the benefits of formal methods are concrete and economically realizable. A very favorable reaction was received from the IBM (now Loral) requirements analysts and senior JSC personnel (Bob Hinson, in particular). They commented that they would like to work with us "to build a different paradigm where engineers write requirements like this before passing the requirements to software development."

4.2.2 Honeywell Navigation Specification

A cooperative research effort was initiated in 1993 with Honeywell Air Transport Systems Division (Phoenix) to study the incorporation of formal methods into the company's software development processes. In the initial project in this effort, NASA Langley funded ORA to identify a component of the Boeing 777 system to which formal specification techniques could be applied, and to develop the formal specifications for that component. ORA, in collaboration with personnel from Langley and Honeywell, chose the navigation subsystem as a suitable application.

Using documents supplied to them by Honeywell, ORA developed a specification that addressed the following aspects of navigation:

- basic mathematical concepts such as functions over the reals, and physical units such as distance, velocity, and acceleration
- definition of objects such as aircraft, radios, sensors, navigation aids, and the navigation database
- definition of algorithms such as complementary filter processing, navigation aid selection, navigation mode selection, and position determination
- relating the mathematical model to Ada by partitioning the system in Ada package specifications, and annotating individual Ada functions and procedures with formal specifications

The specification was done using ORA's Penelope tool.

4.2.3 Verification of Existing Ada Applications Software

Odyssey Research Associates completed two tasks applying their Ada verification tools to aerospace applications. The first task was to verify some utility routines obtained from the NASA Goddard Space Flight Center and the NASA Lewis Research Center using their Ada Verification Tool named Penelope [64]. This task was accomplished in two steps: (1) formal specification of the routines and (2) formal verification of the routines. Both steps uncovered errors [65]. The second task was to formally specify the mode-control panel logic of a Boeing 737 experimental aircraft system using Larch (the specification language used by Penelope) [66].

4.3 Verification of Hardware Devices

Our past research and technology transfer efforts in the area of formal verification of hardware devices are described below.

4.3.1 CSDL Scoreboard Hardware

A joint project between ORA and Charles Stark Draper Laboratory (CSDL) was completed in 1993. NASA Langley and the Army had been funding CSDL to build fault-tolerant computer systems for over two decades. During this project, CSDL became interested in the use of formal methods to increase confidence in their designs. ORA was given the task of formally specifying and verifying a key circuit (called the scoreboard) of the Fault-Tolerant Parallel Processor (FTPP) [67] in Clio [68]. The formal verification uncovered previously unknown design errors. When the scoreboard chip was fabricated, it worked without any error manifestation. It was the first time that CSDL produced a chip that worked "perfectly" on a first fabrication. CSDL credits VHDL-development tools and formal methods for the success.

4.3.2 Asynchronous Communication

CLI developed a formal model of asynchronous communication and demonstrated its utility by formally verifying a widely used protocol for asynchronous communication called the bi-phase mark protocol, also known as "Bi-Φ-M," "FM" or "single density" [69]. It is one of several protocols implemented by microcontrollers such as the Intel 82530 and is used in the Intel 82C501AD Ethernet Serial Interface.

4.3.3 Digital Design Derivation

Funded in part by a NASA Langley Graduate Student Research Program fellowship, Bhaskar Bose developed the Digital Design Derivation system (DDD) and used it to design a verified microprocessor. DDD implements a formal design algebra that allows a designer to transform a formal specification into a correct implementation[50]. Bose formally derived the DDD-FM9001[70] microprocessor from Hunt's top-level specification of the FM9001 microprocessor[71].

5 Some Observations

Some general conclusions can be drawn from the collective experience of the Langley-sponsored projects. First, modern formal specification languages such as PVS, which support higher order logic and a rich type system, provide a means of writing specifications that can be read and understood by engineers. Second, level 1 formal methods can reveal bugs not found by traditional V&V techniques. Levels 2 and 3 can uncover additional errors and provide increased confidence in the correctness of the system design. Third, a sizable upfront cost is associated with transferring formal methods into a commercial company. The two major components of this cost are: (1) Training the industrial experts to use the formal techniques, especially to develop skill in verification, and (2) The formal methods experts must work closely with the industrial team to learn the problem domain in detail in order to develop effective formal methods. In both cases, the government can play a key sponsorship role. It is critical that the government sponsor properly match the industry interested in formal methods to the appropriate formal methods team. Fourth, Finding a person inside the company who is knowledgeable of the problem domain, is a proponent of formal methods, and is an active participant in the project is essential. Projects lacking such an individual have only produced minimal transfer of formal methods technology. Fifth, the formal methods researchers must be willing to adapt their methods to the problem domain rather than fight to change the existing methodologies to conform to their needs. Sixth, without the significant increases in hardware speed, level 3 formal methods would still be impractical. Efficient automation of reasoning can greatly facilitate the useability of a theorem prover.

6 Summary

The NASA Langley program in formal methods has two major goals: (1) develop formal methods technology suitable for a wide range of aerospace designs and (2) facilitate technology transfer by initiating joint projects between formal methods researchers and aerospace industries to apply the results of the research to real systems. Starting in 1991, NASA Langley initiated several aggressive projects designed to move formal methods into productive use in the aerospace community:

- Boeing PIU Project (1991)
- Charles Stark Draper FTPP Scoreboard (1991)
- Allied Signal Hybrid Fault Models (1992)
- Shuttle Tile Project (1992)
- Space Shuttle Jet Select Project (1993)[6]
- Honeywell Navigation (1993)
- Rockwell Collins AAMP5 (1993)
- Honeywell Tablewise (1994)
- Union Switch and Signal (1994)
- Rockwell Collins AAMP-FV (1995)

NASA's program has advanced aerospace-related formal methods in the United States to the point where commercial exploitation of formal methods is near. Our program has driven the development of PVS, the most advanced general-purpose theorem prover in the world [10], and the Odyssey Research Associates VHDL-verification tool. Commercial industry has been anxious to work with our team, although we have not had sufficient resources to work with as many as we would have liked. Nevertheless, we have helped lay the necessary foundation for productive use of formal methods in several companies.

Fundamental research has been performed in the design and verification of a fault-tolerant reliable computing platform that can support real-time control applications. Also much progress has been made in developing and demonstrating formal methods for critical subsystems of the RCP such as clock synchronization, Byzantine agreement, and voting.

References

[1] Nancy G. Leveson, "Software safety: What, why, and how", *Computing Surveys*, vol. 18, June 1986.

[2] Peter G. Neumann, "Illustrative risks to the public in the use of computer systems and related technology", *ACM Software Engineering Notes*, vol. 19, pp. 16–29, Jan. 1994.

[3] Margaret Hamilton, "Zero-defect software: the elusive goal", *IEEE Spectrum*, Mar. 1986.

[4] "Saab blames Gripen crash on software", *Aviation Week & Space Technology*, Feb. 1989.

[5] Lauren Ruth Wiener, *Digital Woes*, Addison-Wesley Publishing Company, 1993, ISBN 0-201-62609-8.

[6] This project was not initiated by Langley, but Langley has been a major participant in it.

[6] W. Wayt Gibbs, "Software's Chronic Crisis", *Scientific American*, pp. 86–95, Sep. 1994.

[7] Michael Rogers and David L. Gonzalez, "Can we trust our software?", *Newsweek*, Jan. 1990.

[8] *Key Technologies For the Year 2000*, National Center for Advanced Technologies, 1250 Eye Street N.W., Washington, D.C. 20005, June 1991.

[9] Michael Schrage, "'When the chips are down' will likely be heard more often in computing", *The Washington Post*, p. B3, December 16, 1994.

[10] Sam Owre, John Rushby, Natarajan Shankar, and Friedrich von Henke, "Formal verification for fault-tolerant architectures: Prolegomena to the design of pvs", *IEEE-TSE*, 1995, To appear.

[11] John Rushby, "Formal methods and digital systems validation for airborne systems", NASA CR-4551, 1993.

[12] John C. Knight and Nancy G. Leveson, "An experimental evaluation of the assumptions of independence in multiversion programming", *IEEE-TSE*, vol. SE-12, pp. 96–109, Jan. 1986.

[13] John. C. Knight and Nancy. G. Leveson, "A reply to the criticisms of the Knight & Leveson experiment", *ACM SIGSOFT Software Engineering Notes*, Jan. 1990.

[14] Ricky W. Butler and George B. Finelli, "The infeasibility of quantifying the reliability of life-critical real-time software", *IEEE-TSE*, vol. 19, pp. 3–12, Jan. 1993.

[15] Geoff Barrett, "Formal methods applied to a floating-point number system", *IEEE-TSE*, vol. 15, pp. 611–621, May 1989.

[16] Iain Houston and Steve King, "CICS project report: Experiences and results from the use of Z in IBM", in *VDM '91: Formal Software Development Methods*, pp. 588–596, Noordwijkerhout, The Netherlands, Oct. 1991. Springer Verlag, Volume 1: Conference Contributions.

[17] Ricky W. Butler and Sally C. Johnson, "Formal methods for life-critical software", *in Computing in Aerospace 9 Conference*, pp. 319–329, San Diego, CA, Oct. 1993.

[18] Ricky W. Butler, (ed.), "NASA formal methods workshop 1990", NASA CP-10052, Nov. 1990.

[19] Sally C. Johnson, C. Michael Holloway, and Ricky W. Butler, "Second NASA formal methods workshop 1992", NASA CP-10110, Nov. 1992.

[20] Computer Resource Management Inc., *Digital Systems Validation Handbook – volume III*, DOT/FAA/CT-88/10, FAA.

[21] David W. Best, Nick M. Mykris Charles E. Kress, Jeffrey D. Russell, and William J. Smith, "An advanced-architecture CMOS/SOS microprocessor", *IEEE Micro*, vol. 2, pp. 11–26, Aug. 1982.

[22] Natarajan Shankar, "Verification of real-time systems using PVS", In Courcoubetis [72], pp. 280–291.

[23] Natarajan Shankar, Sam Owre, and John Rushby, *PVS Tutorial*, Computer Science Laboratory, SRI International, Menlo Park, CA, Feb. 1993, Also appears in Tutorial Notes, *Formal Methods Europe '93: Industrial-Strength Formal Methods*, pages 357–406, Odense, Denmark, April 1993.

[24] Steve Miller and Mandayam Srivas, "Formal verification of the AAMP5 microprocessor: A case study in the industrial use of formal methods", *in WIFT'95 Workshop on Industrial-strength Formal Specification Techniques*, Boca Raton, Florida USA, Apr. 1995, To appear.

[25] John Kelly, et. al., "Formal methods demonstration project for space applications - phase i case study: Space shuttle orbit dap jet", Dec. 1993.

[26] Ricky W. Butler, "NASA Langley's research program in formal methods", *in COMPASS 91*, Gaithersburg, MD, June 1991.

[27] Sally C. Johnson and Ricky W. Butler, "Design for validation", *IEEE Aerospace and Electronics Systems*, pp. 38–43, Jan. 1992.

[28] Sally C. Johnson and Ricky W. Butler, "Design for validation", *in AIAA/IEEE 10th Digital Avionics Systems Conference*, pp. 487–492, Los Angeles, California, Oct. 1991.

[29] Leslie Lamport and P. M. Melliar-Smith, "Synchronizing clocks in the presence of faults", *Journal Of The ACM*, vol. 32, pp. 52–78, Jan. 1985.

[30] Ben L. Di Vito, Ricky W. Butler, and James L. Caldwell, II, "Formal design and verification of a reliable computing platform for real-time control (Phase 1 results)", NASA TM-102716, Oct. 1990.

[31] Ben L. Di Vito, Ricky W. Butler, and James L. Caldwell, "High level design proof of a reliable computing platform", *in Dependable Computing for Critical Applications 2*, Dependable Computing and Fault-Tolerant Systems, pp. 279–306. Springer Verlag, Wien New York, 1992.

[32] Ben L. Di Vito and Ricky W. Butler, "Provable transient recovery for frame-based, fault-tolerant computing systems", *in Real-Time Systems Symposium*, Phoenix, Az, Dec. 1992.

[33] John Rushby, "Formal specification and verification of a fault-masking and transient-recovery model for digital flight-control systems", NASA CR-4384, July 1991.

[34] Ricky W. Butler and Ben L. Di Vito, "Formal design and verification of a reliable computing platform for real-time control (Phase 2 results)", NASA TM-104196, Jan. 1992.

[35] Ben L. Di Vito and Ricky W. Butler, "Formal techniques for synchronized fault-tolerant systems", *in Dependable Computing for Critical Applications 3*, Dependable Computing and Fault-Tolerant Systems, pp. 279–306. Springer Verlag, Wien New York, 1993.

[36] Ricky W. Butler, Ben L. Di Vito, and C. Michael Holloway, "Formal design and verification of a reliable computing platform for real-time control (Phase 3 results)", NASA TM-109140, Aug. 1994.

[37] Leslie Lamport, "Using time instead of timeout for fault-tolerant distributed systems", *ACM TOPLAS*, vol. 6, pp. 254–280, Apr. 1984.

[38] Jack Goldberg et al., "Development and analysis of the software implemented fault-tolerance (SIFT) computer", NASA CR-172146, 1984.

[39] John Rushby and Friedrich von Henke, "Formal verification of a fault-tolerant clock synchronization algorithm", NASA CR-4239, June 1989.

[40] John Rushby and Friedrich von Henke, "Formal verification of algorithms for critical systems", *IEEE-TSE*, vol. 19, pp. 13–23, Jan. 1993.

[41] Fred B. Schneider, "Understanding protocols for Byzantine clock synchronization", Technical Report 87-859, Cornell University, Ithaca, NY, Aug. 1987.

[42] Natarajan Shankar, "Mechanical verification of a schematic Byzantine clock synchronization algorithm", NASA CR-4386, July 1991.

[43] Natarajan Shankar, "Mechanical verification of a generalized protocol for byzantine fault-tolerant clock synchronization", vol. 571 of *Lecture Notes in Computer Science*, pp. 217–236. Springer Verlag, Jan. 1992.

[44] Paul S. Miner, "An extension to schneider's general paradigm for fault-tolerant clock synchronization", NASA TM-107634, Langley Research Center, Hampton, VA, 1992.

[45] William D. Young, "Verifying the interactive convergence clock synchronization algorithm using the boyer-moore theorem prover", NASA CR-189649, Apr. 1992.

[46] Paul S. Miner, "Verification of fault-tolerant clock synchronization systems", NASA TP-3349, Nov. 1993.

[47] Paul S. Miner, "A verified design of a fault-tolerant clock synchronization circuit: Preliminary investigations", NASA TM-107568, Mar. 1992.

[48] Paul S. Miner, Peter A. Padilla, and Wilfredo Torres, "A provably correct design of a fault-tolerant clock synchronization circuit", *in 11th Digital Avionics Systems Conference*, pp. 341–346, Seattle, WA, Oct. 1992.

[49] Paul S. Miner, Shyamsundar Pullela, and Steven D. Johnson, "Interaction of formal design systems in the development of a fault-tolerant clock synchronization circuit", *in 13th Symposium on Reliable Distributed Systems*, pp. 128–137, Dana Point, California, October 1994.

[50] Bhaskar Bose, "DDD - A Transformation system for Digital Design Derivation", Technical Report 331, Indiana University, Computer Science Department, May 1991.

[51] Leslie Lamport, Robert Shostak, and Marshall Pease, "The Byzantine Generals problem", *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382–401, July 1982.

[52] William R. Bevier and William D. Young, "Machine checked proofs of the design and implementation of a fault-tolerant circuit", NASA CR-182099, Nov. 1990.

[53] William R. Bevier and William D. Young, "The proof of correctness of a fault-tolerant circuit design", *in Second IFIP Conference on Dependable Computing For Critical Applications*, pp. 107–114, Tucson, Arizona, Feb. 1991.

[54] J Strother Moore, "Mechanically verified hardware implementing an 8-bit parallel io byzantine agreement processor", NASA CR-189588, Apr. 1992.

[55] John Rushby, "Formal verification of an oral messages algorithm for interactive consistency", NASA CR-189704, Oct. 1992.

[56] Mandayam Srivas and Mark Bickford, "Verification of the FtCayuga fault-tolerant microprocessor system (Volume 1: A case study in theorem prover-based verification)", NASA CR-4381, July 1991.

[57] Mark Bickford and Mandayam Srivas, "Verification of the FtCayuga fault-tolerant microprocessor system (Volume 2: Formal specification and correctness theorems)", NASA CR-187574, July 1991.

[58] Philip Thambidurai and You-Keun Park, "Interactive consistency with multiple failure modes", *in 7th Symposium on Reliable Distributed Systems*, pp. 93–100, Columbus, OH, Oct. 1988.

[59] Patrick Lincoln and John Rushby, "A formally verified algorithm for interactice consistency under a hybrid fault model", NASA CR-4527, July 1993.

[60] Patrick Lincoln and John Rushby, "Formal verification of an algorithm for interactive consistency under a hybrid fault model", In Courcoubetis [72], pp. 292–304.

[61] Patrick Lincoln and John Rushby, "A formally verified algorithm for interactive consistency under a hybrid fault model", *in Fault Tolerant Computing Symposium 23*, pp. 402–411, Toulouse, France, June 1993. IEEE Computer Society.

[62] Patrick Lincoln and John Rushby, "Formal verification of an interactive consistency algorithm for the draper ftp architecture under a hybrid fault model", *in COMPASS 94*, June 1994.

[63] John Rushby, "A formally verified algorithm clock sychronization under a hybrid fault model", *in ACM Principles of Distributed Computing '94*, Aug. 1994.

[64] David Guaspari, "Penelope, an ada verification system", *in Proceedings of Tri-Ada '89*, pp. 216–224, Pittsburgh, PA, Oct. 1989.

[65] Carl T. Eichenlaub, C. Douglas Harper, and Geoffrey Hird, "Using Penelope to assess the correctness of NASA ada software: A demonstration of formal methods as a counterpart to testing", NASA CR-4509, May 1993.

[66] David Guaspari, "Formally specifying the logic of an automatic guidance controller", *in Ada-Europe Conference*, Athens, Greece, May 1991.

[67] Richard E. Harper, Jay H. Lala, and John J. Deyst, "Fault tolerant parallel processor architecture overview", *in Proceedings of the 18th Symposium on Fault Tolerant Computing*, pp. 252–257, 1988.

[68] Mandayam Srivas and Mark Bickford, "Moving formal methods into practice: Verifying the FTPP scoreboard: Phase 1 results", NASA CR-189607, May 1992.

[69] J Strother Moore, "A formal model of asynchronous communication and its use in mechanically verifying a biphase mark protocol", NASA CR-4433, June 1992.

[70] Bhaskar Bose and Steven D. Johnson, "DDD-FM9001: Derivation of a verified microprocessor. an exercise in integrating verification with formal derivation", *Proceedings of IFIP Conference on Correct Hardware Design and Verification Methods*, pp. 191–202. Springer, LNCS 683, 1993.

[71] Warren A. Hunt, "A formal HDL and its use in the FM9001 verification", in C.A.R. Hoare and M.J. Gordon, editors, *Mechanized Reasoning in Hardware Design*. Prentice-Hall, 1992.

[72] Costas Courcoubetis, editor, *Computer Aided Verification, CAV '93*, vol. 697 of *Lecture Notes in Computer Science*, Elounda, Greece, June/July 1993. Springer Verlag.