SAND2011-1595C

A Survey of Formal Verification in Mission-critical, High-consequence Applications

Yalin Hu, Robert C. Armstrong
Sandia National Laboratories
Livermore, California
USA
yhu@sandia.gov
rob@sandia.gov

Abstract— **Formal verification has been used widely in digital system designs and has been demonstrated as an effective and reliable way to ensure functional correctness and hardware reliability. Formal verification has enabled designers to (1) reduce the time and effort allocated to generate vast number of simulation test benches; (2) reduce the time-to-market for designs on critical path; (3) gain high confidence of the design with reasonable verification effort; (4) accomplish increased design densities with emergent silicon process technology, such as 28-nm FPGA devices.**

Based on mathematical theorem proving and model checking, formal verification has shown success in a wide range of industrial products. A few examples include memory controllers, microprocessor, network equipment, medical devices, and embedded software. We survey different formal verification methodologies, including both static and dynamic formal verification methodologies in the context of the design space on which they focus.

This work discusses automated verification strategies of mission-critical, high-consequence FPGA or ASIC designs, as well as embedded software systems. Application areas include aerospace applications, automotive applications, and cryptographic devices and software. These systems have stringent hardware requirement, often involving harsh environment (e.g. radiation). Because of the high-consequence of a fault, such systems must be verified more thoroughly than a pedestrian consumer product. Challenges include highly complex logic and circuit design along with high concurrency, which is an active area of. As a part of this work, we present a collection of various formal verification technologies and attendant case studies.

Keywords- formal verification, formal methods, hardware verification, software verification, model checking, mission-critical, tools, FPGA, ASIC, VHDL, Verilog

I. INTRODUCTION

Design and verification of highly complex, trustworthy hardware and software systems have always been a challenge. There is no doubt that extremely high assurance is expected for mission-critical systems. The use of formal verification technique in such systems has been increasing in recent years. This paper presents a survey on the applied methodology and applications that are targeted.

Mission-critical designs are those that have to work, otherwise a catastrophe could occur. Examples of such systems include: nuclear reactor control systems, automotive safety and control systems, aerospace control systems, spacecraft controllers, military communication systems, etc. [23]. Any fault in a mission-critical system leads to high consequence, and should be avoided with extreme effort. Other safety-critical systems, such as railroad/subway control systems and medical devices, also have very high requirements for reliability and stability. A fault in safety-critical systems could lead to the loss of human life or dramatic damage to the environment. Thus there are some similarities when applying formal verification techniques to mission-critical and safety-critical systems. Most of the time, these systems are required to go through stringent certification and assurance process, which is not required for pedestrian consumer products. Table I lists some of the widely followed safety standards for hardware and software systems.

The complexity of mission-critical systems is continually increasing. In order to meet new challenges the systems need to be very robust and reliable. With the emergent technology in Integrated Circuit (IC) process, Field Programmable Gate Arrays (FPGAs) are becoming more and more popular, both in traditional digital systems designs, and in mission-critical system components [11][24]. Currently FPGAs can be delivered in 28nm node, with programmable logic blocks, configurable memory blocks, complex peripherals, and even embedded hardware Intellectual Property (IP) blocks. FPGAs are attractive because they are flexible, reconfigurable, and easy to design with vendor provided tool software.

To ensure security of mission-critical systems, sensitive Intellectual Properties (IPs) can be protected better with FPGAs compared to custom hardware. It is harder for attackers to target a specific IP or design, if the IP or design is not loaded onto the device until after it is manufactured. One challenge for ensuring system security with FPGA designs is the introduction of vulnerabilities. Often there are design "hooks" which are intended for future enhancement and possible optimization. But they can be used to introduce unintended functionalities, sometimes could be malicious. Other possibilities include design-tool subversion, trustworthiness of foundries, and at the final physical netlist protection.

Standard	Application		Industry	Created By
	HW	SW		
DO-178B		√	Aerospace & Defense	Radio Technical Commission for Aeronautics (RTCA)
DO-254	√		Aerospace & Defense	RTCA
EN 50128		√	Railway Transportation	European Committee for Electrotechnical Standardization (CENELEC)
FSDA	√		Cryptographic Equipment	National Security Agency (NSA)
IEC 60601	√		Medical Equipment	International Electrotechnical Commission (IEC)
IEC 60880		√	Nuclear Power	IEC
IEC 61508	√	√	Heavy Equipment and Energy	IEC
ISO 26262	√	√	Automotive Electronics	International Organization for Standardization (ISO)

Table I
Safety-related hardware/software design standards

Mission-critical systems often need to operate in harsh environment involving extreme temperature and radiation. Such hostile environment makes it infeasible to do a dynamic test of the design. At the same time when silicon becomes denser with smaller transistors, they are more sensitive to lower level of radiation. This trend has lead to the need of more robust radiation-hard and radiation-tolerant designs. Technologies such as Triple Modular Redundancy (TMR) are introduced to mitigate radiation-induced errors. Being able to formally verify designs facing such environment is still a challenge.

In addition to be rad-had or rad-tolerant, mission-critical systems also need to be fault-tolerant under various circumstances. Most of the time the faults are non-deterministic, making exhaustive testing infeasible and verification task harder.

Traditionally, hardware designs are validated through simulation and emulation, while software systems are validated through code reviews and dynamic testing. As a mature technology, a good simulation test bench could demonstrate the presence of a design bug (i.e. assure the design does what it is supposed to do), but can never ensure the absence of a design bug (i.e. assure the design does not do what it is not supposed to do).

Formal verification for both hardware and software systems provides high level of confidence, automation, and efficiency. As an example, NASA [20] highly recommend applying formal methods for safety-critical software development and verification.

II. DESIGN AND VERIFICATION

A typical design flow that involves formal verification is shown in Figure 1. Specifications (system, functional, property) are normally described in plain text along with block diagrams. Implementation is done in two general ways: hardware description language such as Verilog and VHDL, software programming language such as C/C++. Property modeling can be done with formal semantics. The verification framework then generates the result, which can be used to modify the implementation or specification. Mission-critical and safety-critical systems have much rigorous requirement to be satisfiable [1][10][19].

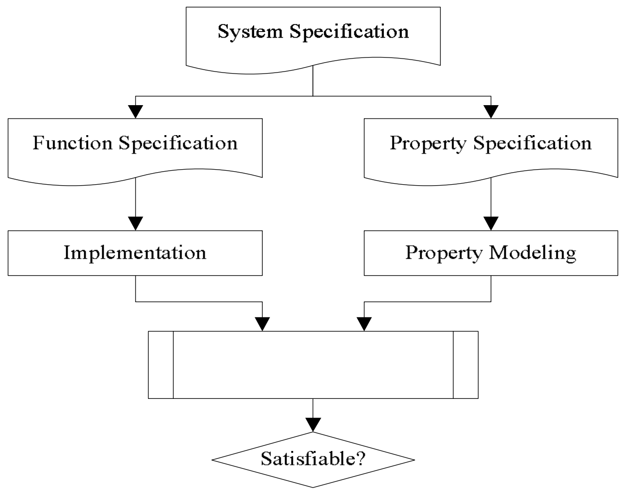

Figure 1. A typical design process with formal verification

Formal Language	Description	Application
Cryptol [2]	Domain Specific Language (DSL)	Cryptography
Esterel [6]	Synchronous language with formal semantics	Aerospace
LOTOS [4]	Language of temporal ordering specification	Communication Protocols
Promela [4][6]	Process meta language	Aerospace, Medical Devices, Spacecraft
SIGNAL [5]	Block-diagram based synchronous language	Real-time System Design
SMV [4][17]	Synchronous language	Rail Transportation

Table II

Surveyed formal language

Formal Framework	Description	Supported Language
Cadence SMV	Deterministic	SMV, Verilog
CADP	Probabilistic	LOTOS
Cryptol Tool	Deterministic	Cryptol language
SCADE	Deterministic	Esterel
NuSMV	Deterministic	SMV
ROMEO	Deterministic	Time Petri Nets
SPIN	Deterministic	Promela

Table III
Surveyed formal framework

III. CASE STUDIES

This section presents several case studies to demonstrate the application of formal method and formal verification for mission-critical and safety-critical systems. There are both hardware and software applications and each one is summarized for their modeling language, formal framework, unique contribution, and the impact on the applications.

A. *FPGA-based Aerospace hydraulic Monitoring System*

Hammarberg and Nadjm-Tehrani [6] published an application of formal verification in an aerospace hydraulic monitoring system. The system detects hydraulic leakage inside a JAS 39 Gripen multi-role aircraft. This is a high-consequence system, because an electrical fault could lead to the complete loss of control of the aircraft in worst case. The co-designed system contains one software component and two FPGA-based hardware components. The purpose of using two separate FPGA devices is to increase redundancy in the system, making it more fault-tolerant.

Traditional Fault Tree Analysis (FTA) was not able to describe such complex system, a formal verification based design model was implemented as shown in Figure **. Esterel Studio provides two model checkers, one based on Binary Decision Diagrams (BDD) and another one based on propositional Satisfiability (SAT). The SAT based solver was chosen for this particular design. The main goals of this verification are (1) verify single fault tolerance of the system, and (2) identify potential double fault combinations. In order to achieve co-design and co-verification, all three components and nets that connect them are modeled in Esterel. The top level structure of a developed verification bench is shown in Figure 2. The highlighted verification bench is written as plug-in modules. These modules are solely for verification purposes, and are ignored during design code generation and system implementation. The output from the verification bench ("Alarm Signal") indicates whether there is a fault detected or not.

Possible hardware faults, such as bit flipping on silicon (FPGA or processor) can be caused by environmental factors, such as radiation, extreme temperature, and sudden power change. A fault switch is inserted to serve as a fault injector. The objective of such fault switch is to indicate whether a formally verified safety-related property would hold if an environment fault presents. An example of environmental fault modeled in this application is the arbitrary malfunction in either of the FPGA devices.

Esterel's built-in model checker does a good job in this application, especially with the support of user-provided constraints. The verification results are impressive by proving: (1) the components do not contain design faults causing violation of the safety property; (2) no combination of the potential faults can cause violation of the property; (3) no single random fault can cause violation of the property; and (4) the only double fault violating the property is when the software component and one of the FPGA component are faulty.

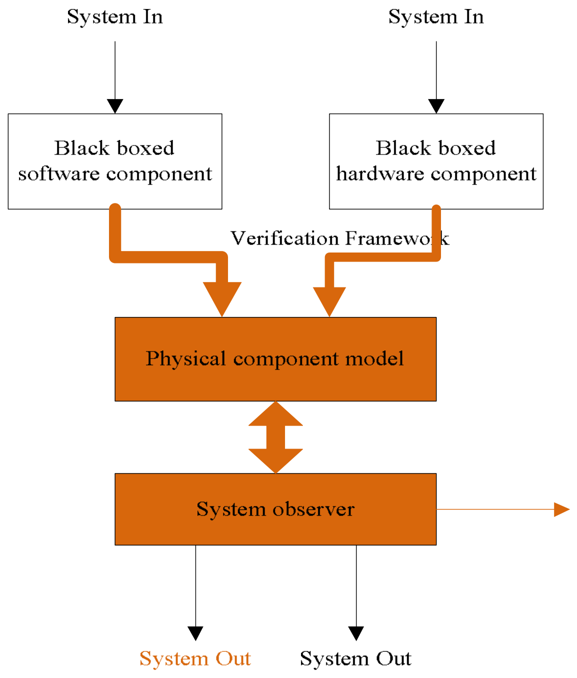

Figure 2. Hardware/Software co-verification model

Another advantage of this approach is the short run time. The model checking takes a few second to run, while a simulation test bench with descent coverage can easily run in hours, even days for such a complex system.

The authors also demonstrated a comparison between manually created and automatically generated VHDL design for another smaller safety-critical application. The example is the PID controller used in a brake control system for an aircraft arrester system. The same design is implemented in two ways: (1) manually created a VHDL design, and (2) automatically generated VHDL code from Esterel model. Both designs are then run through the FPGA design flow (synthesis, place & route, timing analysis). The manual design wins in both area (logic usage on device) and speed (Fmax of the design). However, Esterel generated VHDL design has smaller size (lines of code) in general.

This is a case study that demonstrates a practical design process for mission-critical system. The design is specified at a high abstraction level, which is implementation independent. With the built-in verification bench, it successfully detected random faults that are of high-consequence. The tradeoff is the implementation efficiency, which could lead to the need of a bigger and faster FPGA device. This tradeoff, however, can be easily justified for such applications.

B. *Multi-thread Control Module for Space Craft*

Havelund, Lowry and Penix [8] published a formal analysis case study of a space craft controller. The software to be verified is a component of NASA's Remote Agent (RA), an artificial intelligence (AI) based space craft control system architecture. The module is developed in LISP programming language and is multi-threaded. The Remote Agent itself is a mission-critical application as it is the first AI based software that demonstrated the complete control of a space craft.

SPIN is chosen to be the model checker for this application, as it supports verification of finite state asynchronous process systems. A domain specific language (DSL) named Executive Support language (ESL) is used to specify the bottom layer of the module. The verification scheme is shown in Figure 3. By abstraction, the original LISP program is reduced to a finite state system described in Promela, which is a C-like programming language used by SPIN. This abstraction is a critical step for efficient verification, as it makes feasible to create bounded state space. Two properties are fed into SPIN, described either as Promela assertion or Linear Temporal Logic (LTL) formulae. SPIN is then run to verify if both properties are satisfied.

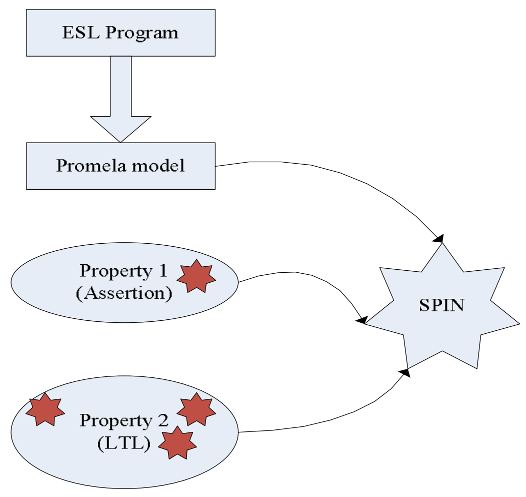

Figure 3. Verification scheme of control module (red stars indicate identified violations)

Outputs from SPIN indicate both properties are not satisfied, with four software errors being identified immediately. With the error trace provided by SPIN for the four bugs, a design flaw (duplicated execution) is also identified. The result is the discovery of five hard-to-find errors, which would manifest themselves only under very particular circumstances involving precise timing. However, these errors are also of very high consequence. A real incident happened during an operation of RA in space, where the thrusting did not turn off as requested, resulting in an immediate action to put the space craft in stand-by mode. This happened when RA was onboard the DEEP_SPACE 1 space craft. It turned out the cause of the failure was an identical error identified by SPIN, but it existed in another module that was not formally analyzed.

This work focused on the development of Promela model. The longest run time of SPIN is less than 1 minute. The result from this work had a major impact on the RA design team, with increased confidence of the delivered software.

This case study demonstrates a very successful application of SPIN's partial order reduction algorithm and state compression.

A related work is reported in [9] that formally analyzes the concurrent software system before and after flight.

Alarm Signal

C. *Model Checking for Fault Tolerant Systems*

Schneider, Easterbrook, Callahan, and Holzmann [22] published a model checking case study to verify a fault-tolerant embedded space craft controller, which is a real-time control system handling critical control sequences. The key contribution of their work is the effective verification based on partial specification. The higher abstraction level is achieved by ignoring unnecessary details, while keeping main properties. Due to the complexity of this application, reducing the state space is crucial to ensure the feasibility of model checking for critical system requirements.

The implementation of this verification scheme is shown in Figure 4. A critical sequence is executed on a deterministic model, with non-deterministic faults injected. Three unrecoverable faults, each indicating a design problem, were identified by this verification scheme.

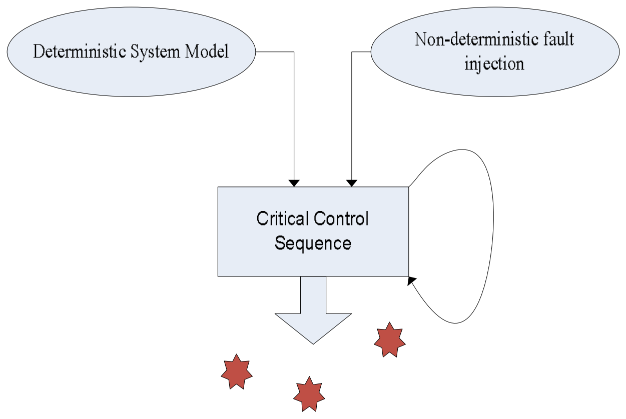
Deterministic System Model
Non-deterministic fault injection
Critical Control Sequence

Figure 4. Fault injection model (red stars indicate identified faults)

With proper modeling, selection of reliable model checker (SPIN), and effective state space reduction, this case study delivered good results in very short run time. The exhaustive examination of selected partial specification runs for about 3 minutes, where as the run time for full specification is estimated to be 10^{12} years. The three design problems identified could lead to potential fault control sequence. Another notable contribution of this case study is the parallel design-verification process, which allows prompt feedback and dynamic modification of both design and specification, as shown in Figure 5.

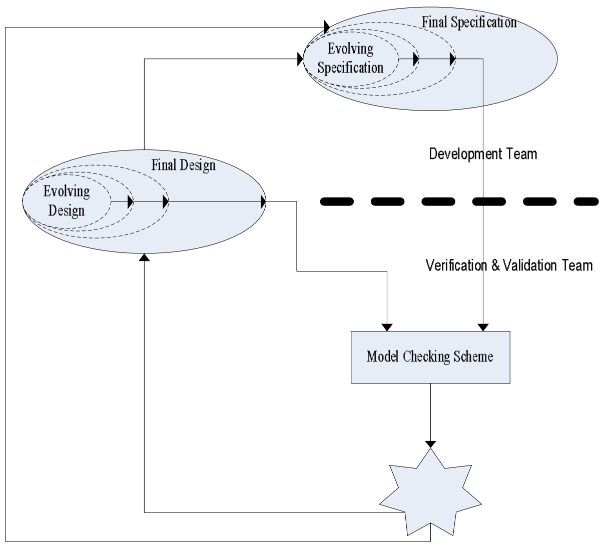

Figure 5. Co-design and verification process

D. Cryptographic Applications

Cryptographic applications require very high level of assurance, performance, reliability, and security. Historically programmable logic has not been widely used because of the challenge to support multiple levels of security and handle isolated redundancy. FPGAs are suitable for implementing cryptographic algorithms because there are a lot of bit-level operations, such as shifting and permutation. With the growing logic density and performance of FPGA devices and development tools [2][7][16], it is now feasible to implement a cryptographic system (even Type I) on a single FPGA chip. However, such designs have to be partitioned in a way that isolated subsystems do not leak information to each other. For example, strong isolation is expected to segregate plain text (red text) and cipher text (black text). The communication between these partitions has to be tightly controlled to meet the National Security Agency's (NSA) Fail Safe Design Assurance (FSDA) requirements.

The primary goal of verifying a cryptographic system is to ensure the risk of compromising its integrity caused by a hardware fault is minimized. Lewis, Hoffman, and Browning [14] published a design and verification flow for implementing a single FPGA-based cryptographic system. This flow leverages a Domain Specific Language (DSL) named Cryptol and tools to support it. Cryptol is a functional description language designed for the NSA as a public standard for cryptographic algorithm specification. It allows the user to create specifications at a much higher level of abstraction compared to structural or behavioral description of digital systems. Even the final implementation is physically on a FPGA, the design process is independent of hardware features and detailed configuration. Compared to any hardware design language (HDL) such as Verilog or VHDL, Cryptol enables the designers to focus on the functional level.

The formal verification feature provided by Cryptol tools focus on equivalence checking. Based on SAT and Satisfiability Modulo Theories (SMT), equivalence checking can be done at various design stages throughout the design process. Similar to Esterel used in an earlier cast study, Cryptol can also generate lower level VHDL designs, which can then be synthesized, placed and routed on a FPGA device. One attractive feature of Cryptol is that the generated VHDL code comes with a formal proof to ensure the functional equivalence. Results have shown that the auto-generated implementations are comparable or better compared to manually written Verilog/VHDL implementation, in terms of area and speed. With the introduction of Signal-Processing Intermediate Representation (SPIR) model, Cryptol provides a nice mixture of easy development at higher level and easy access to lower detailed implementation information. An overview of the design and verification flow for Cryptol is shown in Figure 6.

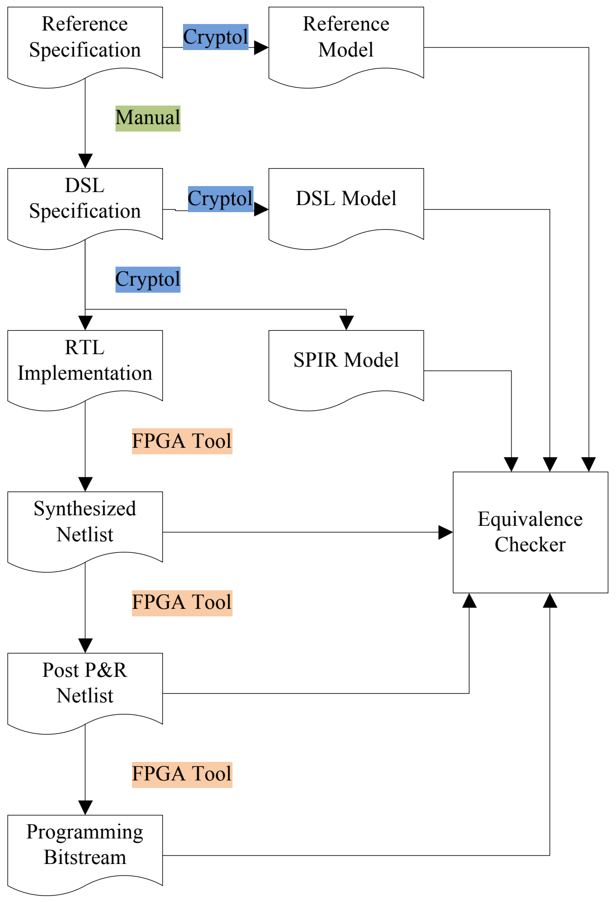

Figure 6. Design/verification flow provided by Cryptol

This case study demonstrates an effective co-design/verification flow for systems with very high-assurance and high-reliability.

E. *Control Software for B-2Test Program*

Chang et al. [3] published a case study in which formal method has made significant contribution to the verification of a mission-critical software system. The targeted application is the Tape Copy and Management System (TCAMS) built for United States Air Force. TCAMS is an important part of the B-2 bomber testing program, which handles enormous amount of flight data during testing. Due to the complexity and extreme requirements of B-2, TCAMS has to be exceptionally reliable.

The overall process of this application is shown in Figure 7. Continuous software verification was made possible through a matrix development model [Tomayko96]. At the requirement analysis stage, formal method was combined with object-oriented analysis to model system specification. At the high level design stage, formal method was used to describe data flow, serial processing ordering, and process parallelization. At the integrated test stage, formal method was used to create test procedures and validation criteria.

Without giving the details of applied formal method and formal verification technique, the authors confirmed that verification of the system was enhanced. The final delivered system achieved exceptional quality and reliability, proven by continuous successful operation upon deployment.

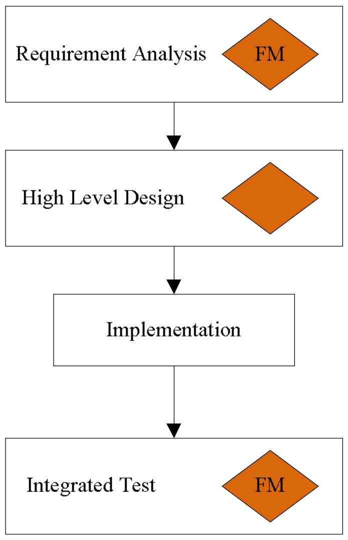

Figure 7. TCAMS design process coupled with formal methods

F. *Formal Modeling and Analysis Military Avionics Systems*

A collaborative research project between the University of South Australia and Australia's Defense Science and Technology Organization aiming at modeling and analyzing avionics mission systems is another success story [21]. The application is an avionics mission system (AMS) for AP-3C Orion maritime surveillance aircraft. The complexity of such systems comes from the large number of hardware and software components, and their integration.

The key contribution of this work is to combine state space methods and Colored Petri Nets (CPN) to reason system properties. Due to the vast number of subsystems and components, complexity can only be managed by higher level of abstraction. CPN was chosen because (1) it provides primitives for modeling concurrency and synchronization; (2) it provides primitives for modeling data manipulation; (3) it is parameterized and can easily be shared for different systems; (4) it supports hierarchical design specification; and (5) it is executable thus can be simulated. In this case study, CPN was used to model different levels of abstraction, allowing formal specification of communications between various subsystems and the avionics bus.

The most challenging tasks for AMS is task scheduling and data transfer management. Task scheduling problem was handled by a state space search approach in this application. If a path from an initial state to a final state is found, then a schedule has been successfully identified. Compared to traditional scheduling algorithms, this approach creates a single model that can be used for both task scheduling and property specification.

All data transfer in this case study happen on a shared data bus, making it critical to ensure the safety and accuracy of data. In this system, data can be transferred between sensors, central control unit, display and storage. The CPN model allows a high level description of the entire data management network.

A remaining challenge for this cast study was the state space explosion problem. As the number of system tasks increase, the state space of the CPN model grows significantly. In the original publication, the author proposed to investigate more advanced methods for reducing the state space in similar models.

Overall, this case study represents an effective formal modeling and analysis approach for a real mission-critical application. The result of this work was the high confidence level of the AP-3C aircraft mission system, which contributes to the aircraft's major missions,, including anti-subsurface/surface warfare, surveillance, search/rescue, and maritime strike.

G. *Aircraft Safety-critical Software*

A recent publication by Yin, Liu, and Su [25] reported a formal verification technique for an aircraft safety-critical software (ASCS) – an aircraft inertia/satellite navigation system. Realizing the general effectiveness of extended finite state machine (EFSM) in formal verification of embedded software systems and its incapability to meet real-time requirements of the ASCS, this work introduced a real-time extension of EFSM, named RT-EFSM.

The developed RT-EFSM model is used to describe the following properties of ASCS: (1) behavior (static and dynamic); (2) real-time characteristics; (3) complex state transition. The same model is also used to solve the state explosion problem and ensure the consistency of ASCS models.

The validation of RT-EFSM involves checking of several critical properties of the model, as shown in Figure 8. Once validated, the model can be used to generate valuable test sequence. A time extended unique input/output (UIO) sequence was introduced to accommodate the real time system. During the test sequence generation, depth-first search tree is constructed for easy traversing and improving test coverage.

The developed formal approach was applied to an aircraft inertia/satellite navigation system. It is reported that the verification methodology is very effective for this application.

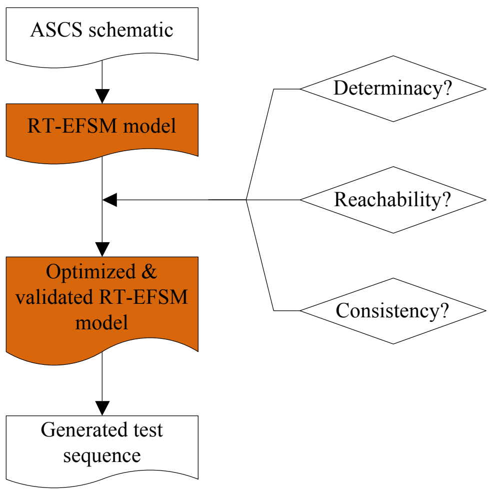

Figure 8. RT-EFSM based verification for ASCS

IV. CONCLUSION

Formal verification has been used in many applications as an alternative to traditional testing approaches – simulation for hardware designs and dynamic testing for software systems. With the extreme requirement of reliability of mission-critical and safety-critical systems, being able to effectively verify the design throughout the design cycle is highly desirable. There is increasing number of published work in applying formal verification to mission-critical and safety-critical systems in recent years.

This paper surveys the recent research and development of formal verification in high-consequence applications. Several case studies are analyzed for their unique application and challenge, applied formal approach, and delivered results. The cases studied include both hardware systems and software systems. For hardware systems, the survey focuses on designs implemented with FPGA, because of its flexibility, reconfigurability, and growing popularity in the targeted applications. The surveyed cases applied various formal methodologies to accommodate different applications, including both equivalence checking and formal model checking.

Additional application of formal verification in safety-critical system include railway interlocking system [15], hybrid emergency control component [13], medical device software system [12] etc.

With the advancing research in formal methods and formal verification technology, more and more successful applications are expected to be published. More advanced formal tools are also expected from the Electronic Design Automation (EDA) industry to further enhance verification. A recent success story is the joint effort between Northrop Grumman Italia and Mentor Graphics to achieve DO-254 compliance [18].

Followed by this survey, innovative mission-critical applications of formal verification will be identified and appropriate technology will be investigated and developed.

ACKNOWLEDGMENT

This work was supported by the Laboratory Directed Research & Development grant from Sandia National Laboratories, US Department of Energy.

REFERENCES

[1] O. Åkerlund, S. Nadjm-Tehrani, and G. Stålmarck "Integration of formal methods into system safety and reliability analysis", Proc. 17th Internatinoal Conference on System Safety, September 1999.

[2] S. Browning, M. Carlsson, L. Erkök, J. Matthews, B. Martin, and S. Weaver, "The next wave", in press.

[3] T. Chang, A. Danylyzsn, S. Norimatsu, J. Rivera, D. Shepard, A. Lattanze, and J. Tomayko, "Continuous verification in mission critical software development", Proc. Thirtieth Hawaii International Conference on System Science, vol.5, pp 273-284, January 1997, doi: 10.1109/HICCS.1997.663184.

[4] E.M. Clarke, O. Grumberg, and D.A. Peled, "Model checking", The MIT Press, 1999.

[5] P. Guernic, T. Gautier, M. Borgne, and C. Maire, "Programming real-time applications with SIGNAL", Proc. IEEE, vol. 79, No. 9, pp. 1321-1336, September 1991.

[6] J. Hammarberg and S. Nadjm-Tehrani, "Formal verification of falut tolerance in safety-critical reconfigurable modules", International Journal on Software Tools for Technology Transfer (STTT) – Special section on formal methods for industrial critical systems, vol. 7, Issue 3, June 2005.

[7] D.S.Hardin, ed. Design and verification of microprocessor systems for high-assurance applications, Springer 2010.

[8] K. Havelund, M. Lowry, and J. Penix, "Formal analysis of a space craft controller using SPIN", IEEE Transactions on Software Engineering, vol. 27, Issue 8, August 2001.

[9] K. Havelund, M. Lowry,S. Park, C. Pecheur, J. Penix, W. Visser, and J. White, "Formal analysis of the remote agent before and after flight", Proc. 5th NASA Langley Formal Methods Workshop, Williamsburg, VA., June 2000.

[10] K.L. Heninger, "Specifying software requiremetns for complex systems: new techniques and their application", IEEE Transactions on Software Engineering, vol. 6, pp. 2-13, 1980.

[11] T. Huffmire, B. Brotherton, T. Sherwood, R. Kastner, T. Levin, T. Nguyen, and C. Irvine, "Managing security in FPGA-based embedded systems", IEEE Design & Test of Computers, vol. 25, Issue 6, pp 590-598, November-December 2008, doi: 10.1109/MDT.2008.166.

[12] P. Jones, R. Jetley, and J. Abraham, "A formal methods-based verification approach to medical device software analysis", Electronic Engineering (EE) Times, 2/9/ 2010.

[13] C. Livadas and N. Lynch, "Formal verification of safety-critical hybrid systems", Proc. 1st International Workshop, Hybris Systems: Computation and Control (HSCC'98), vol. 1386, April 1998.

[14] J.R. Lewis and B. Martin, "Cryptol: high assurance, retargetable crypto development aand validation", IEEE Military Communications Conference, vol. 2, pp. 820-825, October 2003.

[15] W. Ma and X. Hei, "An approach for design and formal verification of safety-critical software", Proc. 2010 International Conference on Computer Application and System Modeling (ICCASM), vol. 4, pp264-268, October 2010, doi: 10.1109/ICCAM.2010.5620084.

[16] M. McLean and J. Moore, "FPGA-based single chip cryptographic solution", Military Embedded Systems, Msrch 2007.

[17] K.L. McMillan, "Symbolic model checking: an approach to the state explosion problem", Kluwer Academics, 1993.

[18] Mentor Graphics' Website published success story, http://www.mentor.com/products/fpga/success/northrop-grumman.

[19] L. Moser and P.M. Melliar-Smith, "Formal verification of safety-critical systems", Software—Practice and Experience, vol. 20, issue 8, pp. 799-821, August 1990.

[20] NASA Software Safety Guidebook[C], NASA-GB-8719.13, NASA 2004.

[21] Z. Qureshi, "Formal modelling and analysis of mission-critical software in military avionics systems", Proc. 11th Australian Workshop on Safety Related Programmable Systems (SCS'06), Conference in Research and Practice in Information Technology, vol. 69, pp. 67-77,

[22] F. Schneider, S. Easterbrook, J. Callahan, and G. Holzmann, "Validating requirements for fault tolerant systems using model checking", Proc. Third International Conference on Requirements Engineering, 1998, pp. 4-13, doi: 10.1109/ICRE.1998.667803.

[23] A. Sutton, "No room for error: creating highly reliable, high-availability FPGA designs", Synopsys Inc. White Paper, November 2010.

[24] P. Taylor, "Using FPGA in mission-critical systems", Electronic Engineering (EE) Times, 12/6/2010.

[25] Y. Yin and B. Liu, "Research on formal verification techniques for aircraft safety-critical software", Journal of Computers, vol. 5, No. 8, pp1152 -1159, August 2010, doi: 10.4304/jcp.5.8.