Title	Verification of automated changeover swithcing unit by model checking
Author(s)	Björkman, Kim; Valkonen, Janne; Ranta, Jukka
Citation	Proceedings of 7th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies, pp. 1719 - 1728
Date	2010
Rights	Copyright © (2010) American Nuclear Society. Reprinted from Proceedings of 7th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies. ISBN 978-0-89448-084-3. This article may be downloaded for personal use only

VERIFICATION OF AUTOMATED CHANGEOVER SWITHCING UNIT BY MODEL CHECKING

Kim Björkman, Janne Valkonen and Jukka Ranta
Technical Research Centre of Finland (VTT)
P.O. Box 1000, FI-02044 VTT, Finland
kim.bjorkman@vtt.fi; janne.valkonen@vtt.fi; jukka.ranta@vtt.fi

ABSTRACT

Along with new NPP designs and modernization projects of ageing NPPs there is an ongoing transition from analogue to digital technology in I&C systems. Programmable digital logic controllers enable more complicated control tasks and, thus, exhaustive verification of such systems by traditional methods is a difficult task. This difficulty is emphasized in cases where digitally implemented systems are combined with old analogue systems. Model checking is a computer-aided method developed for formal verification of correct functioning of a system design by examining all possible behaviors of a model of the system. This paper examines the use of model checking for the verification of a changeover switching unit for a busbar and also summarizes past experiences of utilizing model checking in verification of I&C logic designs. The switching unit is composed of an analogue control logic and a digital malfunction protection relay. The system was analyzed using the NuSMV model checking tool tailored for analysis of systems with a large number of inputs. The case study of this paper expands a series of studies on the applicability of model checking for the verification of NPP automation systems by introducing the problem of state space explosion caused by timing properties. Previous case studies have clearly demonstrated the benefits of model checking in the verification and licensing of digital automation. The analysis of the switching unit demonstrated that model checking is also useful in the verification of I&C systems combining analogue and digital technology, regardless some limitations of the NuSMV tool.

Key Words: safety logic, model checking, verification, changeover switching unit

1 INTRODUCTION

Modern instrumentation and control (I&C) systems are based on programmable digital logic controllers (PLC) that offer system designers a wide range of components for designing complicated control tasks. Traditionally, such systems have been verified by testing and simulation but the transition from analogue to digital technology has created new challenges for safety evaluation. Exhaustive verification by traditional methods is a difficult task, especially in cases where digitally implemented systems are combined with old analogue systems. Traditional system testing and simulation have their advantages and they are useful in many situations but in cases where exhaustive verification with reasonable effort and time is needed none of them alone is suitable. Formal methods are available (see, for example, [14] for an overview) but often they are only used for certain tasks as indicators of possible problems.

Model checking [6] is a computer-aided formal method developed for verification of correct functioning of a system design by examining all possible behaviors of a model of the

system. As in simulation, the models used in model checking describe the behavior of the system design for all sequences of inputs. One of the main differences compared to simulation is that model checkers examine the behavior of the system model for all sequences of input signals. A number of efficient model checking systems have been developed offering analysis tools which are able to determine automatically whether a given state machine model satisfies desired safety properties for realistic designs.

This paper describes the use of NuSMV [11] model checking tool for verification of a changeover switching unit for a busbar and also summarizes past experiences of utilizing model checking in verification of I&C logic designs. The capability and scalability of NuSMV to exhaustively verify a safety logic combining both analogue and digital technology are examined. This case study expands the previous experiences of formal system verification by analyzing a system containing both analogue and digital technology and a state space explosion caused by complex timing functions. Even though some abstractions are made in the model due to complex timing properties, two hidden design errors were discovered that would have been difficult to find by manual inspection or traditional verification techniques.

2 MODEL CHECKING

Model checking [6] is a computer-aided method developed for formal verification of the correct functioning of a system design by examining all possible behaviors of a model of the system. As in simulation, the models used in model checking describe the behavior of the system design for all sequences of inputs. The analysis made with model checking can, at least in principle, be made fully automatic with computer aided tools. Typically, some variant of state machines is used to model the system. The specification is formalized by describing the allowed behaviors of the system in a language understood by the model checking tool, temporal logics being a prime example. The model checker examines the specification against all possible behaviors of the model. If a specification is violated the model checking tool generates automatically a counter-example execution of the system model demonstrating how the property is violated. Otherwise, the system (model) is assumed to be correct with respect to the checked requirements.

Model checking is being utilized in several different domains for software and system design verification. For instance, all major microprocessor manufacturers use model checking techniques to verify their processor designs. In some areas, such as verifying device drivers, model checking real source code has become feasible with new techniques [1]. Model checking has also been applied e.g. in the verification of data communication protocols and real-time controllers.

The system analysis described in this paper utilizes the model checking tool NuSMV [5, 11] which is a state-of-the-art symbolic model checker that supports synchronous state machine models. In NuSMV, the real-time behavior has to be modeled with discrete time steps using explicit counter variables that are incremented at a common clock frequency. NuSMV enables model checking using both Linear Temporal Logic (LTL) and Computational Tree Logic (CTL). The model checking algorithms employed in this work are based on the utilization of Binary Decision Diagrams for symbolic representation and exploration of the state space of the system [10]. For finding errors in larger designs, NuSMV supports also SAT (propositional satisfiability) based bounded model checking [2, 3]. The model checking techniques employed by NuSMV can

handle non-determinism induced by free input variables well, but modeling the real-time aspects can be more challenging due to the inherently discrete time nature of the synchronous state machine model employed by NuSMV.

3 MODEL CHECKING IN I&C DESIGN

3.1 Practical Experiences

The applicability of model checking in verification of NPP digital I&C system designs has been studied by analyzing several industrial cases with two diverse model checking tools NuSMV and UPPAAL [12]. The objective was to evaluate and develop methods based on formal model checking and apply them in the safety analysis of NPP safety automation (I&C). The first case was the analysis of an emergency cooling system of a nuclear reactor [15]. The objective was to verify the correctness of the system's logical functions and test various approaches to modeling. The potential and power of the method were clearly demonstrated and the case gave a good basis for further work.

Another case study concerned an electric arc protection system that is used for protecting switchgear, electrical instrumentation and also people. A basic methodology for modeling such safety instrumented systems was first developed based on hardware model checking techniques and the NuSMV tool [8]. Additionally, an alternative model checking methodology based on timed automata and the UPPAAL tool was investigated [9].

The next case study was a safety-related logic used for stepwise control of an industrial process. The system controls the process towards the normal operating state in case of disturbances. The performance and applicability of two model checking tools, NuSMV and UPPAAL, were analyzed and compared [4]. Both tools were successfully used for verifying basic safety properties of the system and were able to reveal the same hidden design errors. In addition to verifying the correct behavior of the design, the NuSMV tool was used to analyze the fulfillment of the single failure criteria with several different failure models. The case study demonstrated that besides finding design errors, it is possible to determine the fulfillment of single failure criteria with model checking.

An approach for model checking timed embedded software was developed for UPPAAL model checker in the case concerning an UPS (Uninterruptible Power Supply (UPS) [7]. Several failure cases related to the operation of the UPS were investigated. The results of the case study indicate that model checking can be used for verifying the correct operation and finding errors from embedded control software.

The studied cases have clearly demonstrated the power of model checking in verification of I&C systems designs. The method is directly usable for verification designs of safety I&C systems containing tens of inputs and rather complicated and overlapping timing behavior. An advantage of the model checking method compared to traditional testing and simulation activities is that it can provide full coverage for verification in reasonable time [13]. Additionally, model checking enables extensive verification of both positive and negative properties, such as "All system executions lead to a certain system state." or "None of the system executions lead to a certain state.", which is not possible with traditional testing or simulation.

3.2 Methodology

The verification of system designs with model checking includes the following phases: modeling the design, specifying the requirements, running the model checker, and interpreting the results. The system design to be verified has to be captured using the modeling language of the model checking tool. For systematic modeling, it is important to identify the system boundaries and the interface between the system and its environment, and exploit the component structure of the design to create a corresponding modular modeling approach.

The selection of a suitable abstraction level for the model is also an important issue to make model checking feasible. To keep the computing time reasonable irrelevant details of the system need to be abstracted. However, the abstraction level needs to be considered carefully to avoid loosing any relevant system behavior. The abstraction is always a trade-off between the accuracy of the model and the computational efficiency. Verifying whether a system design satisfies a specification is done against an environment model describing how the environment and the system interact. The case studies described in Section 3.1 were analyzed using environment models that allowed the environment to behave quite freely and independently of the system design. This leads to safe model checking results because if the model checking tool determines that the system model satisfies the specification, then this is the case in all kinds of environments and the correct behavior is not based on assumptions on the environment. On the other hand, if the model checking tool generates a counter-example it is quite straightforward to determine whether the counter-example is caused by a too freely behaving environment. In such case, the behavior of the environment can be made more specific.

Requirements are typically given as natural language statements that need to be formalized in the specification language (such as temporal logics) supported by the model checking tool. Requirement formalization can be a challenging task since the requirements are often ambiguous and vague. Given the system model and a requirement formalized for model checker, performing the actual model checking is often the most straightforward part and, thus far, the only phase that is automated. Other phases are still done manually and, thus, may be prone to human errors. In principle, interpreting the results produced by the model checking tool is quite simple, but occasionally counter-example executions demonstrating a violated system design requirement can be quite complex and further tool support may be needed to illustrate the underlying erroneous system behavior.

4 DESCRIPTION OF THE AUTOMATED CHANGEOVER SWITCHING UNIT FOR A BUSBAR

This case study complements the earlier experiences of model checking by analyzing a system that contains both analogue and digital technology and a state space explosion caused by complex timing functions. The analyzed system is called an “Automated changeover switching unit for a busbar” and it is implemented using traditional analogue technology. The purpose of the system is to switch the power feed to an alternative power supply in case of voltage breaks lasting over 1s. The overall system consists of three separate parts (see Figure 1):

1. The process surrounding the switching unit (the system environment).
2. The switching unit (analogue control logic).
3. The malfunction protection relays (digital technology).

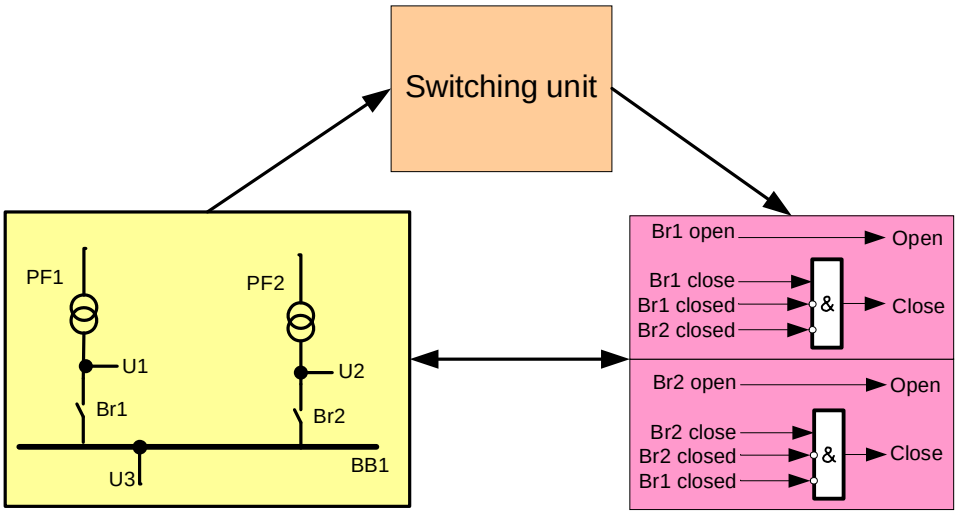

Figure 1. The overall system

The system environment consists of two alternative power feeds PF1 (primary) and PF2 (secondary) that can feed power to the busbar BB1. The changeover switching unit controls the feed breakers Br1 and Br2. In normal power plant operation, the switching unit is passive. In case the voltage of BB1 drops below 65% of the nominal-voltage (U3 < 65%) for 1.6s, the switching unit is triggered.

Figure 2 illustrates a sequence chart describing the functionality of the switching unit. In the event of voltage loss of BB1 both feed breakers are given the “open” command. When both breakers are open, the breaker Br1 is given the “close” command if the voltage of PF1 is higher than 85% of the nominal-voltage (U1 > 85%). If U1 is below 85% and the voltage of PF2 is higher than 85% of the nominal-voltage (U2 > 85%), Br2 is given the “close” command. If the voltage level of both power feeds is below 85%, the switching unit waits until one of the power feeds has the required voltage level and the “close” command is given to the respective breaker.

Before a control command activates a breaker there is a malfunction protection relay ensuring that if one of the breakers is closed, further close commands to any of the two breakers are blocked by the relay. An open command is passed through the relay without any interference.

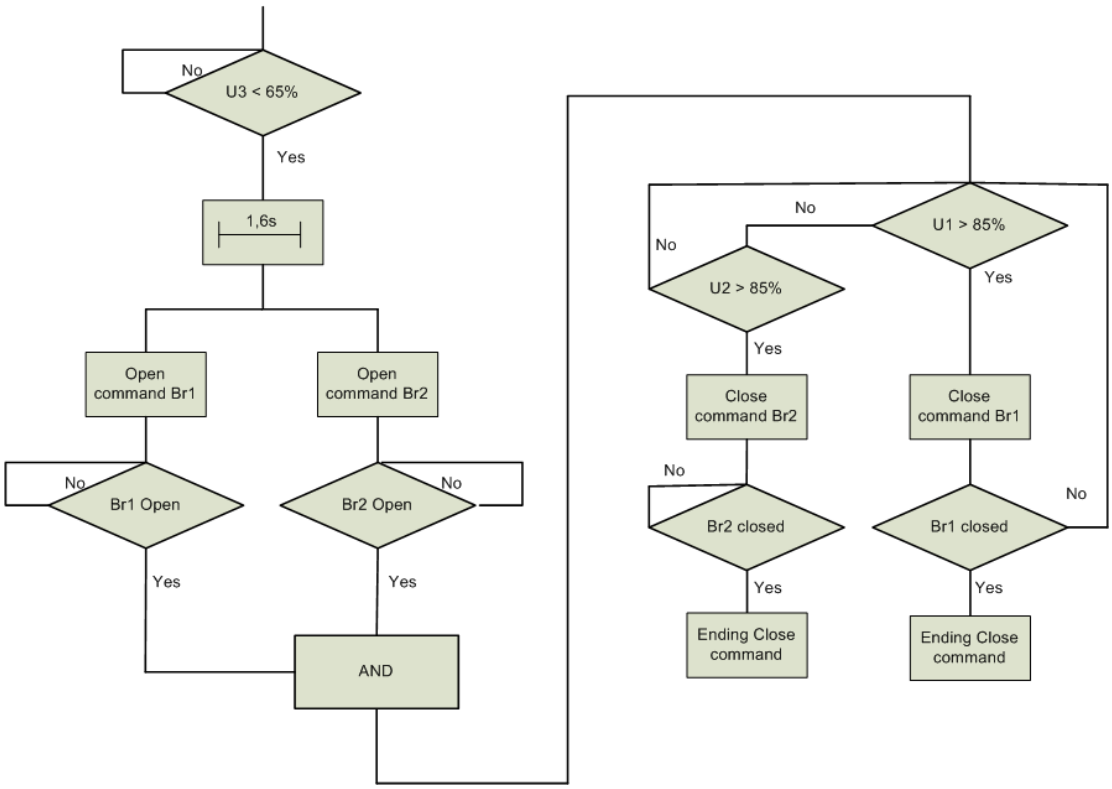

Figure 2 Sequence chart outlining the functionality of the switching unit

5 MODELLING THE AUTOMATED CHANGEOVER SWITCHING UNIT FOR A BUSBAR

5.1 Model of the System

The formal models created for analyzing the automated changeover switching unit are based on the logic diagrams in the system documentation and on the comments by the system experts. The created NuSMV model is composed of a set of modules that are collections of declarations, constraints, and specifications. The system's functional entities, such as the control logic and the malfunction protection relay, were modeled as individual modules. Since a module can contain instances of other modules, a structural hierarchy was constructed.

A module library containing the relevant function blocks was created in the beginning of modeling. These reusable modules were utilized in the implementation of the switching unit logic. The malfunction protection relay was modeled as a black box that functions as realistically as possible. Besides the functionality presented in Section 4, the relay had an arbitrary operation cycle between 700ms and 1200ms.

The environment model was divided into two modules. The functionality of the breakers was modeled in one module and the behavior of the power feeds and the busbar was modeled in another module.

5.2 Timing Issues

The system consists of several timed functions of different lengths. The lengths ranged from 1ms, caused by the inner operation of some logical blocks, to 2s (the length of a time pulse block). Because NuSMV does not support continuous time, the time-dependent components were modeled to operate in discrete fixed-length time steps. The first version of the model included all the original delays without scaling. However, model checking was not feasible because the computation required more memory than available. Thus, the delays had to be scaled down to mitigate the state explosion problem caused by complex timing functions. Additionally, without scaling the manual analysis of the possible counter-examples would have been too laborious.

The objective of scaling was to retain all relevant system behavior in the model while avoiding the state explosion problem. Even though the time behavior in the scaled model is much rougher than in the original model, it can be said with considerable certainty that all the behaviors of the scaled model are included in the analyzed system. However, all the behaviors of the analyzed system may not be included in the scaled model, because some of the shortest delays are omitted.

5.3 Results

The NuSMV model was checked against the following properties:

1. If the voltage level of BB1 is below 65% (U3 < 65%) and in either PF1 or PF2 the voltage level is over 85% (U1 > 85% or U2 > 85%) (can alter at each time step), then eventually the voltage level of BB1 is no longer below 65%.
2. The breakers can never be closed at the same time.
3. The breakers cannot receive close and open commands at the same time.
4. If the voltage level of BB1 is over 65%, no actions are taken.

Two possible errors were found during the analysis of the system model. Property 1 is violated if a powerless power feed is connected to the busbar, in which case the breakers cannot be reopened. Figure 3 illustrates a part of the control logic that concerns the opening of the breakers. A powerless power feed could be connected to the busbar if, e.g. the respective power feed loses its power during the closing of the breaker. If a powerless power feed is connected to the busbar, the breakers cannot be reopened, because a “close” command combined with respective closed breaker generates a signal that sets a set-reset flip-flop (SR1), which in turn resets the set-reset flip-flop (SR2) controlling the opening sequence (as well as flip-flops controlling the closing sequences). The set-reset flip-flop SR1 is reset by the signal indicating a high voltage level in busbar BB1. Additionally, setting SR2 requires that the output of the pulse block (PB1) is 1, which is triggered by a rising edge of the signal indicating low voltage in BB1. Because the low voltage signal is continuously 1, PB1 cannot be retriggered.

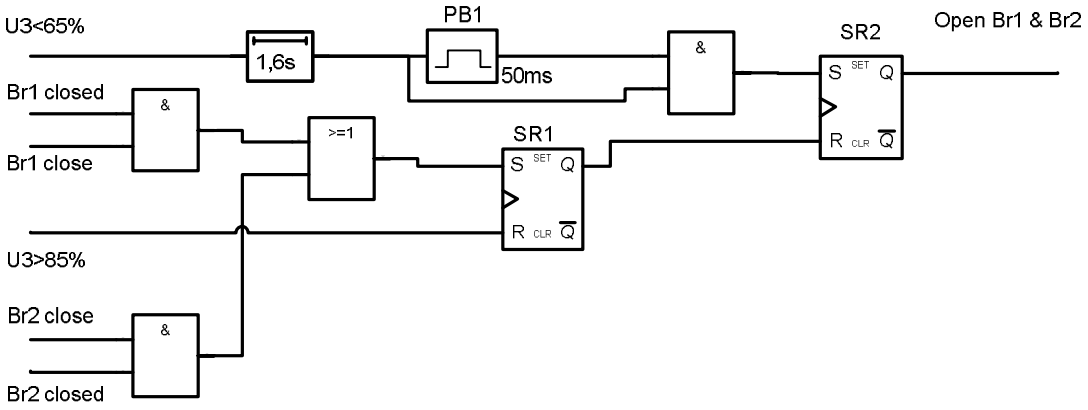

Figure 3. Part of the switching unit logic controlling the opening of the breakers.

Additionally, property 2 was violated. Also in this case a powerless feed is being connected to the busbar but this time the breaker to be closed has to be Br1 and the closing of the breaker has to last over 2s. If the closing of the breaker takes under 2s the consequence will be the same as for the violation of property 1. The design was found to operate correctly with respect to properties 3 and 4.

The situations where the properties 1 and 2 are violated require highly rare conditions where several independent failures occur within a very short time frame of less than 2s. However, this does not diminish the value of the evidence provided by model checking. It just demonstrates that such improbable, strictly timing related events are nearly impossible to discover by traditional methods. Testing such cases with the real system is also practically impossible.

The model checking was carried out with NuSMV version 2.4.3 on a PC with 16GB of RAM and an Intel Xeon CPU X5460 processor running at 3.16GHz. The model checking times were between 2s and 18h depending on the checked property. The total number of states in the model was $2 \cdot 10^{28}$, out of which $2 \cdot 10^{9}$ were reachable. The time step utilized was 2ms. When using a 1ms time step, the model-checking times were unreasonably long.

6 CONCLUSIONS

The several analyzed case studies have shown that model checking is a suitable and useful method for supporting safety-critical software and system development and verification. The main benefits of model checking are achieved at the design phase of the development process. Model checking facilitates the definition of interesting test cases for the actual system testing and it often reveals design errors that could easily be left hidden with only traditional verification methods or be found much later in the development cycle causing extra effort and cost. However, integrating model checking into the system verification process sets high requirements on the correctness of the used tools, modeling abstractions, and the checked specifications.

Traditional verification methods like testing and simulation have their advantages but none of them alone is suitable for exhaustive verification with reasonable effort. Model checking provides full coverage for verification by examining all the possible behaviors of the system model. Creating and modifying models for model checkers is rather straightforward and fast. The

purpose of model checking is not to replace testing and simulation but to complement and support them. Each of the methods has its own strengths and weaknesses and there is a proper role for all of them in the overall system development and verification process.

The analyzed automated changeover switching unit contained several timing functions of different lengths varying in the range of 1ms to 2s. NuSMV model checking tool does not support continuous time, thus time dependent functions had to be modeled to operate in discrete time with fixed length steps. To make model checking feasible with NuSMV, the lengths of the timing functions had to be decreased by scaling them down. The scaled model may contain some behaviors that are impossible in the real system, which is not a problem because the possible unrealistic behavior can be eliminated by examining counter-examples and modifying the model.

The analysis showed how challenging model checking timed systems is and that the limits of the NuSMV model checking tool can be easily reached when the model contains several timed functions of various lengths. This brings out the need for simplification of the model and scaling of the timed functions. It was already realized in the previous case studies that NuSMV performs well when the large state space of the model is caused by a high number of input variables. For the research group, this case study was the first one having the state explosion problem caused by the timed functions and solved with scaling.

Model checking large and complex systems where complexity is caused by high number of input signals or complex timing functions, is still quite challenging. New approaches are required to tackle with these problems. One possible solution is to develop a modular model checking methodology, in which a larger system is split into more manageable sub-systems. The verification of the whole system could be performed by combining the results obtained by model checking the separate sub-systems. If necessary, the different sub-systems could then be analyzed with different model checking techniques. UPPAAL, for instance, could be utilized if complex timing functions are present in the sub-system, and NuSMV could be used if complexity is caused by a large number of input variables.

7 REFERENCES

1. T. Ball, B. Cook, V. Levin, and S. Rajamani, “SLAM and static driver verifier: Technology transfer of formal methods inside Microsoft”, In Proceedings of the 4th International Conference on Integrated Formal Methods (IFM 2004), volume 2999 of Lecture Notes in Computer Science, pages 1-20. Springer-Verlag.
2. A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, “Symbolic model checking without BDDs”, In Proc. of the Fifth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS’99), (1999).
3. A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan, “Linear Encodings of Bounded LTL Model Checking”, Logical Methods in Computer Science 2(5:5):1–64, 2006.
4. K. Björkman, J. Frits, J. Valkonen, J. Lahtinen, K. Heljanko, I. Niemelä and J.J. Hämäläinen, Verification of Safety Logic Designs by Model Checking, Sixth American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies, NPIC&HMIT 2009, Knoxville, Tennessee, April 5-9, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL 2009.

5. R. Cavada, A. Cimatti, C. A. Jochim, G. Keighren, E. Olivetti, M. Pistore, M., Roveri, and A. Tchaltsev, “NuSMV 2.4 User Manual”, CMU and ITC-irst (2005).
6. E. M. Clarke, Jr., O. Grumberg, and D. A. Peled, Model Checking, The MIT Press, (1999).
7. J. Frits, Model Checking Embedded Control Software. Research Report TKK-ICS-R28, Aalto University School of Science and Technology, Department of Information and Computer Science, Espoo, Finland, March 2010.
8. M. Koskimies, Applying model checking to analysing safety instrumented systems. Research Report TKK-ICS-R5, Helsinki University of Technology, Department of Information and Computer Science, Espoo, Finland, June 2008.
9. J. Lahtinen, Model checking timed safety instrumented systems. Research Report TKK-ICS-R3, Helsinki University of Technology, Department of Information and Computer Science, Espoo, Finland, June 2008.
10. K. L. McMillan, “Symbolic Model Checking”, Kluwer Academic Publ., (1993).
11. NuSMV Model Checker v.2.4.3. http://nusmv.irst.itc.it/ (2010).
12. UPPAAL integrated tool environment v. 4.0.6, http://www.uppaal.com/ (2010).
13. J. Valkonen, K. Björkman, J. Frits, I. Niemelä, ”Model Checking Methodology for Verification of Safety Logics”, In Proceedings of the 6th International Conference on Safety of Industrial Automated Systems (SIAS 2010) Tampere, June 14-15, 2010.
14. J. Valkonen, I. Karanta, M. Koskimies, K. Heljanko, I. Niemelä, D. Sheridan, R. E. Bloomfield, “NPP Safety Automation Systems Analysis - State of the Art”, VTT Working Papers 94, VTT Technical Research Centre of Finland, Espoo, Finland, April 2008, 62 p. (2008).
15. J. Valkonen, V. Pettersson, K. Björkman, J.-E. Holmberg, M. Koskimies, K. Heljanko, and I. Niemelä, Model-Based Analysis of an Arc Protection and an Emergency Cooling System - MODSAFE 2007 Working Report. VTT Working Papers 93, VTT Technical Research Centre of Finland, Espoo, Finland, February 2008, 51 p.