VTT Technical Research Centre of Finland

Model-checking I&C logics — insights from over a decade of projects in Finland

Pakonen, Antti

Published in:
12th Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies (NPIC&HMIT 2021)

DOI:
10.13182/T124-34322

Published: 01/06/2021

Document Version
Publisher's final version

MODEL-CHECKING I&C LOGICS — INSIGHTS FROM OVER A DECADE OF PROJECTS IN FINLAND

Antti Pakonen
VTT Technical Research Centre of Finland Ltd
P.O. Box 1000, FI-02044 VTT, Finland
antti.pakonen@vtt.fi

dx.doi.org/10.13182/T124-34322

ABSTRACT

Model checking is a formal, computer-assisted verification method, used to prove that a model of a (hardware or software) system fulfills stated properties. In Finland, VTT has successfully applied the method in practical nuclear projects since 2008, verifying instrumentation and control (I&C) design in the Olkiluoto 3 EPR, Loviisa 1&2 I&C renewal, and Hanhikivi-1 projects. By 2020, we have detected 66 confirmed design issues, in some cases leading to design changes. In this paper, we first discuss the practical impact model checking has had in Finnish NPP projects. We then discuss statistics and characteristics of the detected design issues. Of particular interest is the fact that the method can in practice reveal I&C application logic design issues that could lead to spurious actuation of I&C. With the permission of our clients, we have collected hundreds of industry project models, thousands of formal properties, the detected issues, as well as the counterexamples that revealed them. We have used this data for different experiments in the Finnish Research Programme on Nuclear Power Plant Safety (SAFIR). In the second part of the paper, we present an overview of the past and current research activities, to demonstrate the state of the art. A recent example is the work on verifying the I&C application logics' tolerance to failures of the underlying hardware architecture.

Key Words: model checking, formal verification, I&C software, failure tolerance

1 INTRODUCTION

Methods and techniques used for the verification and validation (V&V) of digital I&C system application logics each have their own strengths and weaknesses. With many approaches, an inherent limitation is that not all possible scenarios are considered—test coverage, for example, is incomplete. It is particularly challenging to demonstrate that something *unwanted*, like a spurious actuation, *cannot* occur.

Formal verification methods rely on mathematical proof. Tools like model checkers can calculate that an unwanted state or execution path is not possible, at least for a model of the system of interest. The limitations have to do with the difficulty of mastering the formal languages (particularly the languages for specifying formal properties—the wanted, or unwanted, behavior), lack of domain-specific and user-friendly tools, and the overall perceived cost.

In Finland, VTT has used model checking in practical nuclear industry projects since 2008. To date, we have identified 66 confirmed design issues in I&C logics, in some cases leading to the systems' redesign. While the probability and the safety relevance of the issues varies, both the regulator and the utilities have found the method so useful that it has become a standard industry practice in Finland.

Our work is based on the free open-source model checker NuSMV [1], and the graphical frontend called MODCHK [2] developed by VTT and the utility Fortum. In our research activities with Aalto

University, we try to broaden the scope in which model checking can be used, and look for ways of making the overall work process easier.

2 MODEL CHECKING

Model checking [3] is a formal verification method, where a software tool called a model checker analyzes whether a stated property holds for model of a system. If the model checker finds an execution path that violates the property, the path is returned to the analyst as a counterexample scenario. Review of the counterexample can reveal a design issue (or a human error made by the analyst in specifying either the model or the property).

The analysis is exhaustive—the model checker explores all the reachable states of the system model. The challenge is state space explosion, where the number of states to enumerate through becomes too enormous to handle computationally [3]. Symbolic model checkers [1] avoid the problem using Binary Decision Diagrams to avoid explicit state enumeration. Bounded model checking (BMC) [4] makes the analysis even faster, by using Boolean satisfiability (SAT) solvers, and limiting the search depth (in terms of length of the state transition sequences) [4]. The proof obtained by BMC is therefore not as strong.

The system model is typically some form of state machine, but depending on the model checker in use, the model is described in either finite or infinite-state terms. We have mostly worked with the symbolic model checker NuSMV [1], which is based on discrete time processing (and finite-state models), but tools also exist that support real-time analysis or probabilistic model checking.

What is common to the different tools, is that the formal properties are usually expressed using some form of temporal logic language, e.g., Linear Temporal Logic (LTL), Computation Tree Logic (CTL) [3], or Property Specification Language (PSL) [5]. LTL and CTL use temporal operators [3]:

- **X** p : p is true in the next state of the path.
- **G** p : p is true at every state of the path.
- **F** p : p is true at some future state on the path.
- p **U** q : q is true at some future state, and at every preceding state on the path, p is true.

Past LTL operators [6], if supported, are:

- **Y** p : p holds in the previous state on the path. (**Y** p is false in the initial state.)
- **Z** p is otherwise equivalent to **Y** p, but true in the initial state.
- **H** p : p is true at every preceding (and the current) state on the path.
- **O** p : p is true at some past (or the current) state on the path.
- p **S** q : q is true at some past state, and for every state that has then followed on the path, p has been true.

CTL uses branching time, adding the path quantifiers **A** (“for all execution paths”) and **E** (“for some execution path”).

PSL [5] is an extension of LTL, aiming at better human readability. An “LTL style” of PSL called Sequential Regular Expressions (SERE) is convenient for describing multi-cycle behavior. For example the property “starting from the cycle where SET changes to true, ACT shall be true for three cycles” can be written as:

```
always {!SET;SET} |-> {ACT[*3]}!
```

In the next section, we describe the applications of model checking in the Finnish nuclear industry. The use of model checking for nuclear I&C design verification has also been studied in Hungary [7], South Korea [8], and at CERN [9].

3 PRACTICAL PROJECTS IN FINLAND

In this section, we describe customer projects VTT has carried out in the Finnish nuclear industry. Every project has been based on the NuSMV model checker [1]. We have used the graphical frontend MODCHK [2] since 2014. The currently used modelling conventions, work processes and tools are described in [2].

In addition to the projects listed below, VTT has verified rail traffic control logics for the Finnish company Mipro.

3.1 Olkiluoto 3 EPR

Olkiluoto 3 is a 1600 MW EPR in Eurajoki, Finland, owned by the utility TVO. The Finnish government granted an operating license in March 2019, and preparation for fuel loading is ongoing.

In successive projects between 2008 and 2019, on commission from the Finnish Radiation and Nuclear Safety Authority (STUK), VTT has used model checking to evaluate OL3 systems. The work has focused on selected functions of the Protection System (PS), and the Priority and Actuator Control System (PACS). PS is a digital I&C system based on Framatome's Teleperm XS (TXS) technology. PACS is based on Field Programmable Gate Array (FPGA) technology [10]. VTT modelled the application logics of both PS and PACS, and verified how the two systems work together.

According to STUK, "model checking is a very effective method to evaluate complex I&C functions and find design defects that may not be practically possible to find in test field or plant simulator because of the limited possibilities to test input signal and timing combinations. Even when defects do not have direct safety significance, they may give a good impression on the quality of system development processes. [10]"

3.2 Loviisa VVER I&C renewal

The utility Fortum operates two pressurized water reactors of the type VVER-440 in Loviisa, Finland. The reactors started operation in 1977 and 1980. In 2014, Fortum commissioned an I&C renewal project (called ELSA) from Rolls-Royce. The project was completed in 2018, with the new I&C systems based on Spinline technology.

During the ELSA project, on commission from Fortum, VTT has verified selected I&C functions of the Reactor Trip System (RTS), Reactor Power Control System (RPCS), Reactor Power Limitation System (RPLS), Preventive Actuation and Indication System (PAIS), Preventive Protection System (PPS), Neutron Flux Measurement System (NFS), and the functional design of the Manual Backup System (MBS) (functional design). The evaluated systems belong to Finnish safety classes SC2 and SC3, with the exception of the non-safety system RPCS. For the analysis on PPS, VTT also verified how the Rolls-Royce's Spinline based systems operate together with already installed systems developed by another vendor. Based on the results, design modifications have been made to RPCS and RTS.

In Fortum's view, "model checking is seen truly beneficial in nuclear I&C projects, where implementing design changes during or after commissioning is not an easy task. With formal verification, more deficiencies can be found already during the design phase. With conventional I&C, errors can be corrected more or less on-the-fly. In the nuclear domain, all design changes need to be approved internally and major changes to safety classified I&C by the regulator. The obvious benefit of model checking is that through exhaustive analysis, it is possible to find very rare transitional states that occur during a short period of time. With dynamic testing (e.g., running test scripts during the Factory Acceptance Test (FAT)), such rare situations are often not tested. [10]"

3.3 Hanhikivi 1 functional architecture

The utility Fennovoima has submitted a construction license application for an AES-2006 type pressurized water reactor, to be built in Pyhäjoki, Finland by RAOS Project Oy, a subsidiary of Rosatom. The upper-level safety engineering documentation contains the functional architecture, which defines and describes the safety functions of the plant in an early life-cycle stage. The functionality of the safety functions is presented using platform independent function block diagrams (developed by Atomproekt, another subsidiary of Rosatom). The diagrams resemble the diagrams used by I&C vendors, and also serve as input for the I&C architecture design, and later, actual I&C system design.

For successful safety I&C design, it is essential that the functional architecture is of high quality. The architecture defines the functionality to be implemented, and any changes in the functionality during later design states could cause significant delay for the project. Late changes in the functional architecture would lead to further changes in the detailed I&C design already underway, and therefore also additional V&V effort.

On commission from Fennovoima, VTT has verified successive revisions of the Hanhikivi 1 functional architecture. Model checking has revealed design issues in the safety functions' design. The issues deal with ambiguity in the specification, and potential scenarios that could result in spurious actuation, contradictory actuation commands, or in general, incorrect response to input. According to Fennovoima, solving the issues now in early stage will have a positive impact in later stages. Fennovoima also expects that model checking will be used to assess the eventual detailed design of safety classified I&C.

3.4 Detected design issues

To date, in all the above-mentioned nuclear industry projects combined, VTT has detected 66 confirmed design issues with model checking.

What is especially notable, is that 22 (33%) of the issues deal with *spurious actuation* of I&C. Since the checked formal properties can also address unwanted behavior, the analyst can write properties that specifically address spurious actuation [2]. Given the risks associated with spurious failures, and the difficulty in otherwise excluding them, the benefits of model checking in this regard are obvious.

In 5 (8%) of the issues, the I&C logic is permanently frozen to some output state. In 8 (12 %) cases, the issue is caused by the interaction of several I&C systems. In 20 (30%) cases, the issue involved very exact (millisecond scale) timing of events. In 18 (27%) cases, one primary cause was improper or ill-timed operator behavior. In general, the scenarios that the analysis reveals—especially if the target system has already undergone basic V&V—are often unlikely, complicated, and/or counter-intuitive.

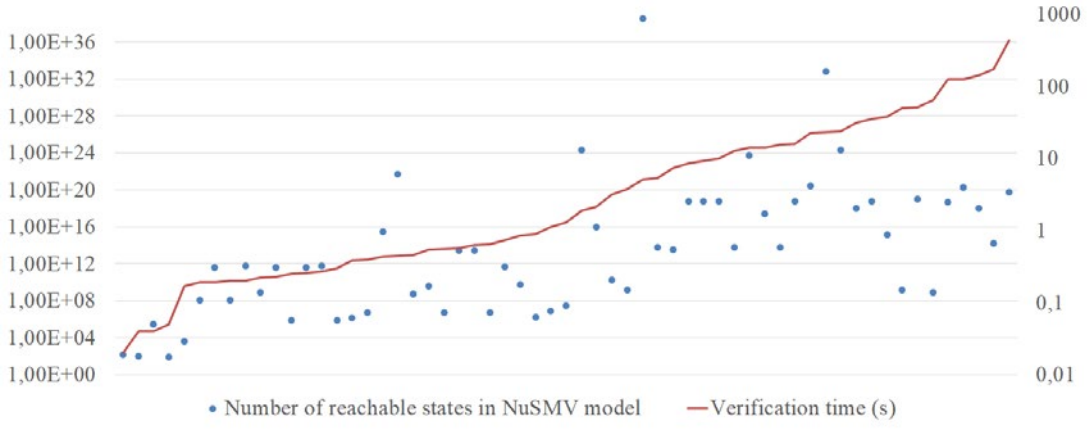

Figure 1. The number of the reachable states in NuSMV models, and the verification time (for the failing property) for 59 of the 66 detected design issues

In Figure 1, we show (1) the number of reachable states in the NuSMV model, and (2) the time it took for NuSMV to verify the failing property and produce the counterexample, for 59 of the 66 detected design issues. We have omitted the seven issues we detected using BMC [4], in those cases the number of reachable states varied between $2.31 \cdot 10^{13}$ and $4.52 \cdot 10^{28}$, and the analysis took from 0.2 to 2.3 seconds. (While there is some correlation between the state space and the analysis time, the number of reachable states alone is not a sufficient measure of complexity in model checking.) The analysis times are based on running NuSMV on an Intel Core i7-6600U CPU with a clock rate of 2.6 GHz.

In Table 1, we list the types of oft-occurring formal properties we used to detect the issues. The letters p, q and r stand for propositions in Boolean algebra. Property types 6 and 9 are particularly useful for analyzing spurious actuation [2]. The PSL constructs 4 and 5 are convenient when addressing sequencing and timing [11]. The CTL property 7 reveals scenarios where the logic freezes permanently to state ¬p.

Table 1. Types of properties that have revealed design issues in industrial projects

#	Generalized property type	Number of revealed issues	%
1	**G** (p → q)	13	20 %
2	**G** ((p ∧ **X** q) → **X** r)	10	15 %
3	**G** ¬p	8	12 %
4	never {*SERE*}	8	12 %
5	always {*SERE*} |-> {*SERE*}!	6	9 %
6	**G** (p → **O** q)	5	8 %
7	**AG EF** p	2	3 %
8	**G** p → **G** ¬q	2	3 %
9	**G** p → **G** (q → **O** r)	2	3 %
	other	10	15 %

The safety relevance of the issues varies. In VTT's own analyses, we assessed potential end effects ranging from negligible (e.g., "periodic test fails", "short equipment transient") to serious (e.g., "spurious actuation may lead to leak of radionuclides", "spurious actuation reduces options for controlling accident", "safety function lost") [12].

3.5 Practical example

As an example, let us consider an issue detected in a safety function containing a set point selection logic. Below, we have significantly simplified the logic, and otherwise tried to mask its origin. The originally reported scenario was also more complicated, and revealed design issues beyond incorrect set point selection.

The simplified logic shown in Figure 2 selects the control set point to use while the safety function is active. If the temperature is high, the logic selects the high set point. If the temperature is lower ("NORMAL TEMP."), the logic switches to a low set point. Once the temperature has lowered, the operator can also reset the safety function.

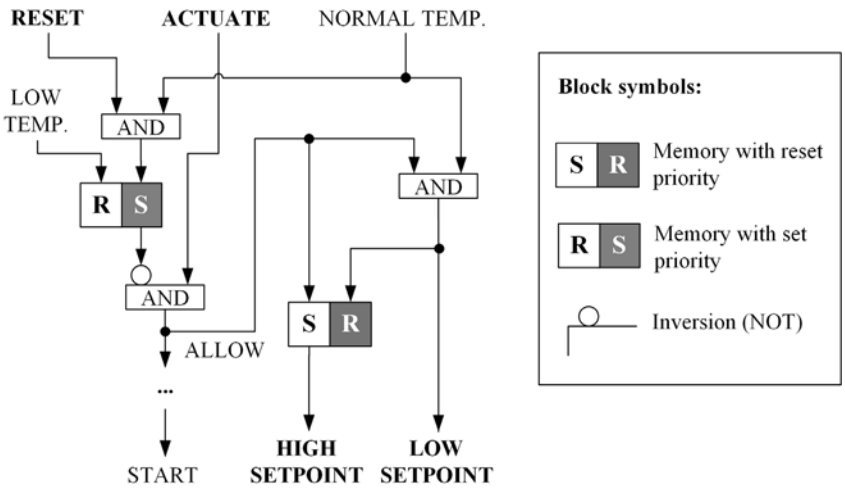

Figure 2. Exemplar logic—setpoint selection

One requirement for the logic is therefore: "When the temperature has returned to normal level, the operator can reset the set point to the low value." One option to state the requirement as an LTL property is: **G** (((ACTUATE ∧ HIGH_SETPOINT ∧ ¬RESET) ∧ **X** (NORMAL_TEMP ∧ RESET)) → **X** LOW_SETPOINT).

NuSMV produces a counterexample, which—after modified by the analyst to a more probable scenario—we illustrate in Figure 3. The key element is that the operator uses the reset option while the temperature is still high (which would be a counterintuitive and questionable act). Once the temperature returns to the normal value, the reset and temperature signals do reset the "ALLOW" signal, but resetting "ALLOW" also means that the high set point is *not* reset (see Step 2 in Figure 3). From this point on, the high set point can only be reset after the temperature has dropped to a low value (and the safety function still active or re-activated).

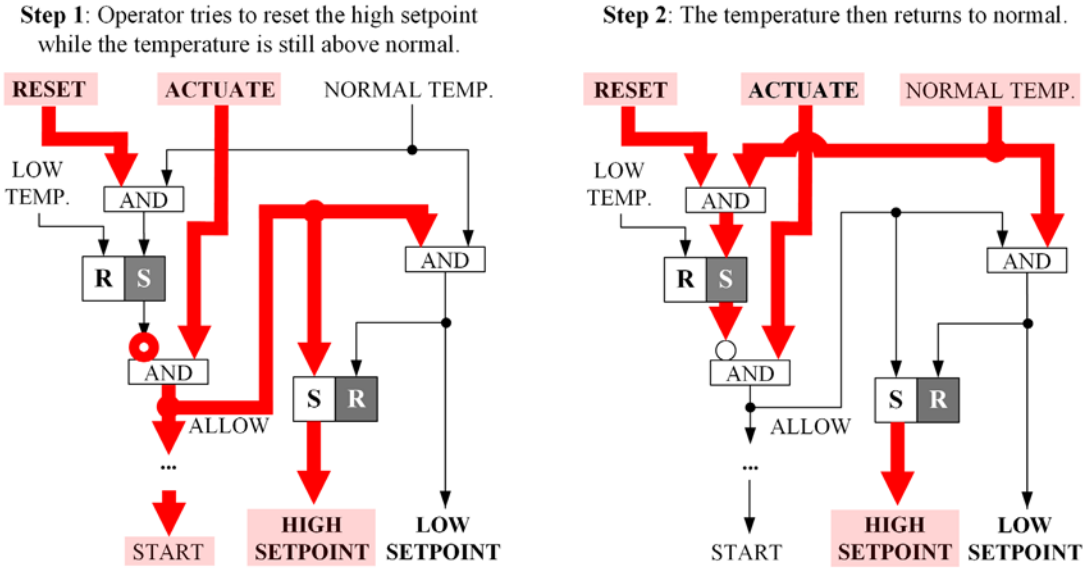

Figure 3. A counterexample scenario that reveals a design issue

The original model contained logic from several related safety functions, and consisted of 56 function blocks. The logic collected to Figure 2 is actually scattered throughout the different functions. (We also changed the property from the original. The analyst detected the issue when verifying a requirement that did not directly address the functionality discussed here.) The original NuSMV model contained $6.60 \cdot 10^{13}$ reachable states, and the analysis times varied between 5 and 15 seconds depending on property.

Other practical examples, similarly masked, can be found in, e.g., [2], [10] and [13].

4 RESEARCH TOPICS

The Finnish Research Programme on Nuclear Power Plant Safety (SAFIR) develops and creates expertise, experimental facilities, and methods for solving safety issues. In the last two four-year programme periods (SAFIR2018 and SAFIR2022), VTT and Aalto University have cooperated to develop methods and prototype tools for model-checking nuclear I&C systems. In this section, we list some of the topics we have jointly worked on.

4.1 Hardware failure tolerance assessment

VTT's practical projects so far have focused on the I&C application logic as if it were operating "in a vacuum". In reality, the underlying I&C hardware is subject to ageing and failure. Since the logics are meant to be failure-tolerant, it would be important to also verify that aspect, but previous attempts to include the hardware failures have resulted in excessive computational complexity [2]. By focusing on single failure tolerance, utilizing symmetry (see Figure 4), and simplifying the failure modelling approach, we were able to inject the failures in a straightforward way. We then experimented by injecting the failures to models created in the practical industry projects, and found that the computational overhead from single failure modelling was negligible [2].

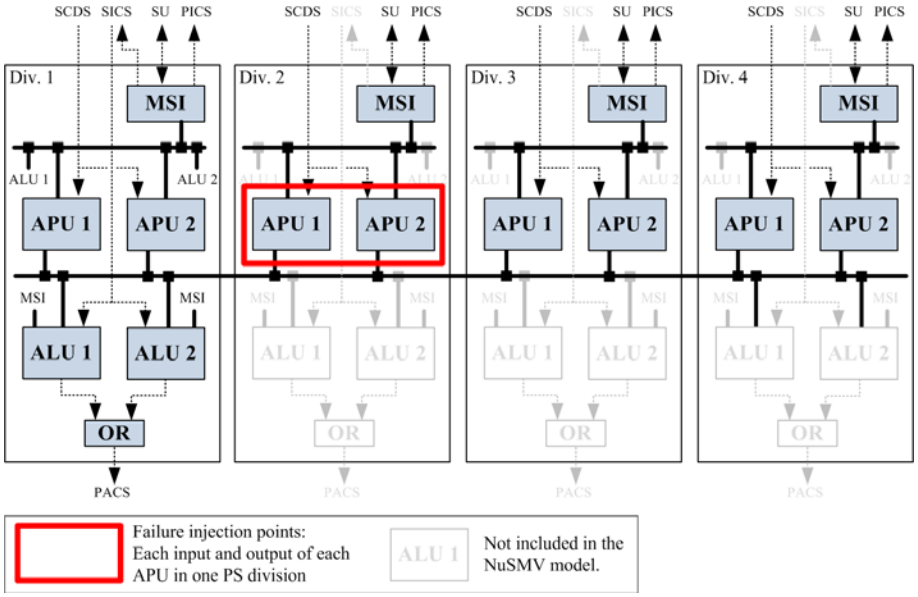

Figure 4. When analyzing the single failure tolerance of a many-redundant system, symmetry helps in simplifying the hardware failure model [2].

In [14], we broadened the scope to the I&C architecture level, and included systems from three Defence-in-Depth (DiD) levels in our (semi-fictitious) case study. We then considered tolerance for common-cause failure (CCF) in the lower DiD level functions, in addition to the single failure criterion for the system of interest.

4.2 User-friendly property specification

One of the foremost challenges in model checking is the difficulty in mastering the languages used for formal property specification. Techniques suggested to make the work easier include textual patterns, timing diagrams, sequence charts, directed graphs, and special function block diagrams. In [11], we collected over a thousand properties specified by VTT experts in the practical industry projects, and then reviewed the different approaches based on that data.

Although the silver bullet remains elusive, we arrived at some conclusions. First, a relatively small number of specification patterns can cover a large part of the specification needs (see also Table 1). Second, any language aimed at the verification of I&C systems needs to handle timing and sequencing conveniently. Third, many of the visual languages—while perhaps more familiar to I&C engineers—are better suited at capturing scenarios than properties. In all, a small set of patterns, combined with basic PSL skills, seems like a reasonable option to learn when getting started.

4.3 Counterexample explanation

MODCHK visualizes the counterexamples by animating the function block diagram [2]. This "model view" is obviously useful, but does not specifically draw attention to where, when or why the property violation actually occurs. If the counterexample is long (tens of time steps) and the model is complex, it may take the analyst some effort to pinpoint the root of the cause.

In [13], we experimented with two techniques to assist in counterexample interpretation. First, in addition to animating the model view, we animated the property (using colors to show the subformula values on each counterexample step). Second, we used a causality-based approach to highlight variables deemed important for understanding the counterexample. Using a prototype tool (see Figure 5), we then evaluated the techniques by analyzing the counterexamples from all the 43 design issues VTT had by then detected.

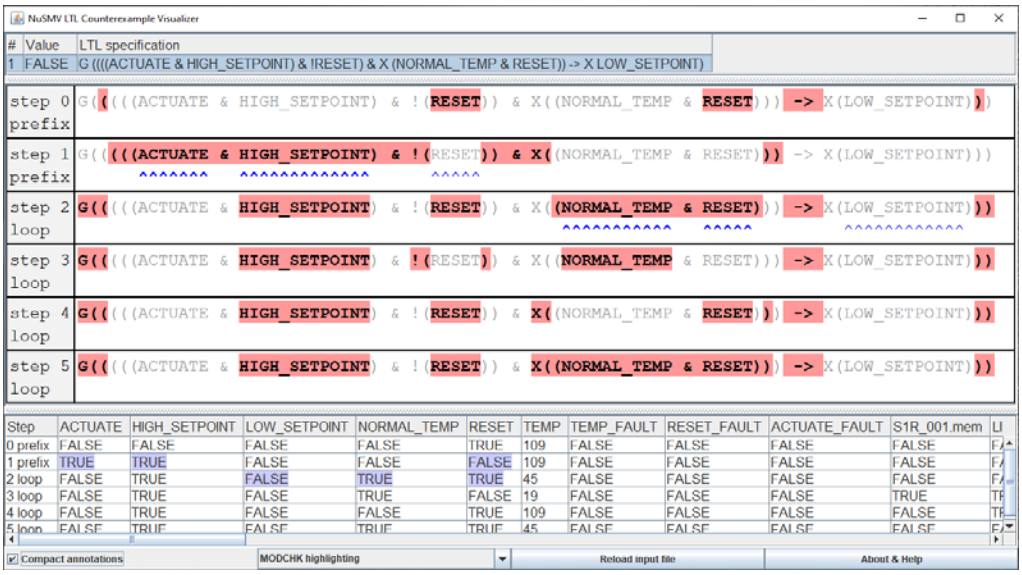

Figure 5. Counterexample Visualizer animates the property (red highlight for true), and highlights important variables (blue arrows).

(The example in Figure 5 is based on the logic from section 3.5. In this counterexample, the "RESET" signal activates on the exact processing cycle when the "NORMAL_TEMP" is also set. In Figure 3, the analyst has modified the scenario to a more likely one that does not require exact timing.)

In [15], we demonstrate another prototype tool that allows the analyst to click on any signal in the model view, and the tool then highlights paths in the function block diagram that are important for understanding why the selected signal has the value that is has on that counterexample step.

4.4 Tool benchmarking

We have found that the symbolic verification algorithms used by NuSMV are more efficient for our purposes than the explicit-state algorithms used in tools like Spin. NuSMV models are based on discrete state transitions, but continuous time modelling tools like Uppaal seem to be unable to handle logics of similar complexity.

In [16], we extended the NuSMV models we work with to timed models in nuXmv [17], in an attempt to account for communication delay in a distributed I&C architecture. The resulting complexity meant that BMC [4] was the only viable option for verifying the timed properties.

Recently, we have also experimented with nuXmv's infinite-domain algorithms, which in practice means support for real-valued variables. Using nuXmv, the analyst does not then have to spend effort simplifying the I&C functions containing complex analog processing logic. Our experiments have shown that nuXmv outperforms NuSMV, but some useful types of properties are not supported for the infinite-domain case.

5 CONCLUSIONS

In Finland, model checking has successfully been used, for over a decade, in every major project involving digital I&C systems in nuclear power plants. Our experience proves that the method reveals real, sometimes serious design issues that would otherwise be hard to detect. In our models of the application software, we can also account for hardware failures, and analyze failure tolerance. With the right tools, constant re-evaluation of designs (as they evolve) is fast.

We hope our example helps pave the way for the broader acceptance and adoption of formal verification methods. Work is still needed to further develop the tools, methods and practices. However, we are already way past the argument of whether or not the benefits justify the cost.

6 ACKNOWLEDGMENTS

Research activities presented in this paper have been funded by the Finnish Research Programme on Nuclear Power Plant Safety 2019-2022 (SAFIR2022). We thank our customers (Fennovoima, Fortum and STUK) for letting us use confidential data for research purposes.

7 REFERENCES

1. A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella, "NuSMV 2: An opensource tool for symbolic model checking," *Proceedings of the 2002 International Conference on Computer Aided Verification (CAV)*, LNCS vol. 2404, pp. 359-364 (2002). https://doi.org/10.1007/3-540-45657-0_29
2. A. Pakonen, I. Buzhinsky and K. Björkman, "Model checking reveals design issues leading to spurious actuation of nuclear instrumentation and control systems," *Reliability Engineering & System Safety*, **Volume 205**, 107237 (2021). https://doi.org/10.1016/j.ress.2020.107237
3. E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, MIT Press, Cambridge, MA (1999).

4. E. Clarke, A. Biere, R. Raimi and Y. Zhu, "Bounded model checking using satisfiability solving," *Formal Methods in System Design*, Volume 19, pp. 7-34 (2001). https://doi.org/10.1023/A:1011276507260 .
5. C. Eisner and D. Fisman, *A practical introduction to PSL*, Springer, Boston, MA (2006). https://doi.org/10.1007/978-0-387-36123-9
6. M. Benedetti and C. Cimatti, "Bounded Model Checking for Past LTL", *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2003)*, LNCS vol. 2619, pp. 18-33 (2003). https://doi.org/10.1007/3-540-36577-X_3
7. E. Németh, T. Bartha, "Formal verification of safety functions by reinterpretation of functional block based specifications," *Proceedings of the 2008 International Workshop on Formal Methods for Industrial Critical Systems (FMCIS)*, LNCS vol. 5569, pp. 199-214 (2008). https://doi.org/10.1007/978-3-642-03240-0_17
8. J. Yoo, S. Cha and E. Jee, "Verification of PLC Programs Written in FBD with VIS", *Nuclear Engineering and Technology*, **Volume 41**, pp. 79-90 (2009). https://doi.org/10.5516/NET.2009.41.1.079
9. B. F. Adiego, D. Darvas, E. B. Viñuela, J. Tournier, S, Bliudze, J. O. Blech, et al. "Applying model checking to industrial-sized PLC programs,", *IEEE Transactions on Industrial Informatics*, **Volume 11**, pp. 1400-1410 (2015). https://doi.org/10.1109/TII.2015.2489184
10. A. Pakonen, T. Tahvonen, M. Hartikainen and M. Pihlanko, "Practical applications of model checking in the Finnish nuclear industry," *10th International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC & HMIT 2017)*, San Francisco, CA, USA, June 11-15, pp.1342-1352 (2017).
11. A. Pakonen, C. Pang, I. Buzhinsky and V. Vyatkin, "User-friendly formal specification languages — conclusions drawn from industrial experience on model checking," *Proceedings of the 21st IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2016)*, Berlin, Germany, Sep. 6-9, pp. 1-8 (2016). https://doi.org/10.1109/ETFA.2016.7733717
12. A. Helminen and A. Pakonen, "Potential applications of model checking in probabilistic risk assessments," VTT Research Report VTT-R-00017-20, https://cris.vtt.fi/files/27299692/VTT_R_00017_20.pdf (2020).
13. A. Pakonen, I. Buzhinsky and V. Vyatkin, "Counterexample visualization and explanation for function block diagrams," *Proceedings of 16th International Conference on Industrial Informatics (INDIN 2018)*, Porto, Portugal, July 18-20, pp.747-753 (2018). https://doi.org/10.1109/INDIN.2018.8472025
14. I. Buzhinsky and A. Pakonen, "Symmetry Breaking in Model Checking of Fault-Tolerant Nuclear Instrumentation and Control Systems," *IEEE Access*, **Volume 8**, pp.197684-197694 (2020). https://doi.org/10.1109/ACCESS.2020.3034799
15. P. Ovsiannikova, I. Buzhinsky, A. Pakonen and V. Vyatkin, "Oeritte: user-friendly counterexample explanation for model checking," *IEEE Access*, in press, (2021).
16. I. Buzhinsky and A. Pakonen, "Timed model checking of fault-tolerant nuclear I&C systems," *Proceedings of the 18th IEEE International Conference on Industrial Informatics (INDIN 2020)*, Warwick, UK. July 20-23, pp. 159-164 (2020).
17. R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri and S. Tonetta, "The nuXmv symbolic model checker," *Proceedings of the 2014 International Conference on Computer Aided Verification (CAV)*, LNCS vol. 8559, pp. 334-342 (2014). https://doi.org/10.1007/978-3-319-08867-9_22