PRACTICAL APPLICATIONS OF MODEL CHECKING IN THE FINNISH NUCLEAR INDUSTRY

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

Topi Tahvonen
Fennovoima Ltd
Salmisaarenaukio 1, FI-00180, Helsinki, Finland
topi.tahvonen@fennovoima.fi

Markus Hartikainen, Mikko Pihlanko
Fortum Power and Heat
P.O. Box 100, FI-00048 Fortum, Finland
markus.hartikainen@fortum.com, mikko.pihlanko@fortum.com

ABSTRACT

Model checking is a powerful, formal, computer-assisted verification method that can be used to prove that a model of a (hardware or software) system fulfills stated properties. When used right, model checking can prove the correctness of instrumentation and control (I&C) system application logic, be it a software or a field-programmable gate array (FPGA) based design. The verified properties can also address unwanted functionality, making model checking a very effective method for analyzing spurious actuation. Despite the benefits, model checking is still not widely adopted in the verification of I&C systems, one exception being the Finnish nuclear industry. Since 2008, VTT has applied model checking in practical customer work related to the Olkiluoto 3 EPR, Loviisa 1&2 VVER-440, and Hanhikivi 1 AES-2006 nuclear power plants, on commission from either the regulator or the utilities. In this paper, we look at how the method has been used in each of these three projects. We also introduce a user-friendly, graphical modelling tool called MODCHK. We then present an overall view of the design issues detected during nine years of customer work, and one practical example. The 100% coverage means that the design errors that are found often relate to scenarios that are otherwise hard to account for (e.g., exact timing of events, or improper operator actions).

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

1 INTRODUCTION

Formal methods can be used to systematically demonstrate the syntactic correctness of software specifications [1]. Through mathematical analysis, it can be proven that a piece of code is a correct translation of its specification, and that an unsafe state is not reachable [2][3]. According to IEC 61508-3, the use of formal methods is highly recommended at the highest safety integrity level (SIL 4), and recommended on SIL 3 and 2 [4]. Still, industrial acceptance of such methods has been slow, apparently due to the difficulty in mastering these methods, and the lack of tool support [3].

In the Finnish nuclear industry, a formal verification method called *model checking* has been used since 2008 to evaluate instrumentation and control (I&C) system application logics. VTT has evaluated

both software and field-programmable gate array (FPGA) based designs, in the context of both new-builds and an I&C renewal project, on commission from either the regulator or the utilities.

In this paper, we first briefly introduce model checking and the tools used in VTT's practical work, and then look at how the method has been used in the context of Olkiluoto 3, Loviisa 1& 2, and Hanhikivi 1 nuclear power plants (NPP). Finally, we present an overall view on the identified design issues, which highlights how effective model checking is in analyzing scenarios that methods such as testing or simulation can easily miss.

2 I&C MODEL CHECKING

2.1 Model Checking

Model checking [2] is a powerful formal verification method, where a software tool called a *model checker* is used to verify that a finite-state model (of a hardware or a software system) satisfied given properties. The analysis is automatic, fast, and *exhaustive* — covering all possible model behaviors. If a behavior that is contrary to a property is found, it is returned to the user as a *counterexample*. Analyzing the counterexample can then reveal a design issue. Since the analysis is exhaustive, it can reveal design errors that are easily missed using more traditional V&V methods such as testing, simulation, or deductive reasoning, for which 100% coverage is often elusive. The properties can also stipulate that something "bad" should *not* happen, which means that model checking can be used to also address spurious actuation [5].

The verified properties are specified using temporal logic languages such as Linear Temporal Logic (LTL) or Computational Tree Logic (CTL) [2]. Temporal logic describes sequences of transitions between system states, using *temporal operators*, in addition to Boolean algebra. Some of the temporal operators of LTL are listed in Table I. The basic LTL operators all refer to future system states, but it is sometimes more convenient to refer to past states [6]. Examples of LTL properties can be found in Section 4.

Table I. Exemplar temporal operators in LTL

Op.	Semantics
X *p*	"next": *p* is true in the next state.
G *p*	"globally": *p* is true at every state.
F *p*	"finally": *p* is true at some future state.
p **U** *q*	"until": *q* is true at some future state, and at every preceding state, *p* is true.
Y *p*	"yesterday": *p* is true in the previous state.
O *p*	"once": *p* is true at some past state, or the current state.

Property specification is a challenging and error-prone task [7]. To alleviate the issue, several user-friendly formal specification languages have been proposed, many of them visual. In [7], some techniques are evaluated against I&C domain specific criteria.

2.2 Model Checking as a Verification Method for Nuclear I&C

In addition to VTT's work in Finland (which we describe in Section 3), there are a few practical examples of model checking having been used to verify nuclear I&C designs. In the early 1990s, a type of formal verification method was used successfully — although with great effort and limited tool support — to analyze the software design of the Trip Computers of Darlington Nuclear Generating Station in Ontario, Canada [8]. In Korea, researchers have applied model checking to verify function block based software of the APR-1400 reactor protection system [9]. In Hungary, model checking has been used to verify the function block diagrams for the PRISE (Primary-to-Secondary leaking) logic of the Paks NPP [10]. At the European Organization for Nuclear Research (CERN), programmable logic controller (PLC) software written in Structured Text (ST) and Sequential Flow Chart (SFC) has also been verified [11]. In [12], a

design process is described wherein model checking was used to verify the software design of a Steam Generator Pressure Controller and a Primary Heat Transport Pressure Controller.

The consensus among several international nuclear regulators [1] is that there are many issues that should be considered when applying formal methods. In [13], we address the common positions on formal methods listed in [1] item by item, and discuss the I&C specific rationale, challenges, and limitations for the application of model checking in detail.

2.3 MODCHK — A Practical Tool for I&C Design Verification

A lot of the research efforts on I&C software model checking have focused on the automatic generation of the model, based on standard PLC programming languages like the IEC 61131-3 [14][15]. However, in the nuclear domain, the major system vendors (like Areva for TXS and Rolls-Royce for Spinline) use vendor-specific, non-standard blocks, and the source code for the elementary blocks in not available.

VTT has developed a graphical tool [14] called MODCHK for verifying function block based application logics. Since the elementary function block library is constructed manually (based on, e.g., functional block descriptions), MODCHK can be used for the analysis of different vendor-specific logics. Signal validity processing is also included in the models, as it is used in systems like TXS and Spinline [5]. Composite function blocks can also be specified, which is particularly useful for the analysis of systems with many redundant divisions. For now, the verified properties are input using a simple text editor.

Once the model has been constructed and the properties written, MODCHK creates the necessary input files for the NuSMV 2.6.0 model checker [16], runs the analysis, and displays the results. Counterexamples output by NuSMV are visualized using 2D animation that the analyst can play back and forth. Different line colors and thicknesses are used to visualize binary signal values, dashed lines indicate invalid data, and numerical monitors display analogue values (see Figure I). With a graphical, function block based representation, interpretation of the counterexamples is easy.

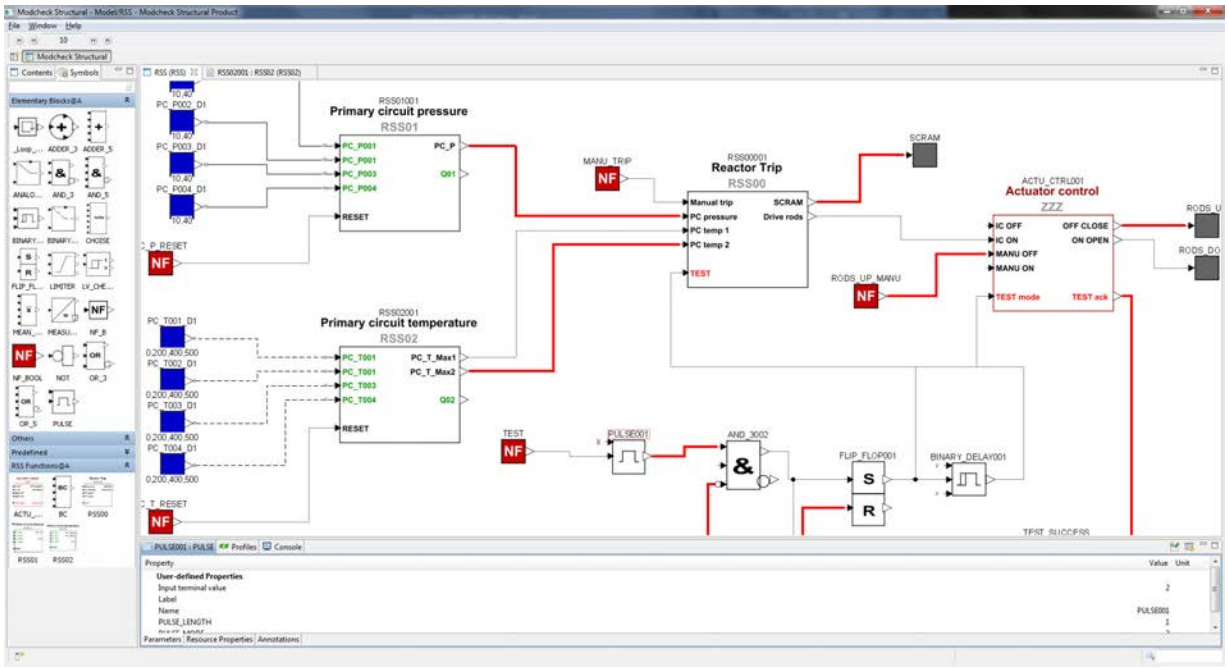

Figure I. MODCHK allows for graphical modelling and counterexample visualization for function block diagrams (fictitious example).

Used in VTT's customer projects since 2014, MODCHK has made practical work more efficient than before. Still, some usability issues need to be addressed before the tool can be distributed.

3 PRACTICAL APPLICATION IN FINNISH NUCLEAR PROJECTS

3.1 Olkiluoto 3 NPP

Olkiluoto 3 is a 1600 MW EPR under construction in Eurajoki, Finland, for the utility TVO. The contractor of the project is Areva. Since 2008, VTT has evaluated OL3 systems using model checking, on commission from the Finnish Radiation and Nuclear Safety Authority (STUK). The analysis has covered selected safety functions of the Protection System (PS), which based on Areva's Teleperm XS (TXS) technology. VTT has modelled the PS application software, expressed as function block diagrams.

In the same context, in 2015, VTT also evaluated the Priority and Actuator Control System (PACS), which is based on FPGA technology. In the overall I&C architecture, the PACS stands between the actuators and main I&C systems. To introduce diversity, the OL3 PACS uses two different TXS modules, the AV42 and the SPLM1-PC11 [17]. In both devices, Programmable Logic Devices (PLD) take care of:

- Acquisition and prioritization of safety-related commands
- Command output and command termination based on travel limit and/or torque limit switch check-back signals from the actuators
- Output of signals to lamps on I&C panels
- Test logic, including lamp test

Some characteristics of both modules can be found in Table II.

Table II. Analysis of FPGA based components of OL3 PACS [17]

TXS Component	PLD I/O #	PLD specification language	NuSMV model states	Analysis times per property
AV42	60	function blocks	~10^{14}	<1s — minutes
SPLM1-PC11	~40	VHDL	~10^{20}	seconds

The PLD logic of the SPLM1-PC11 consists of ca. 2500 lines of VHDL code. For verification, the code was translated into NuSMV input language manually, which was a rather straightforward if repetitive task. The PLD logic of AV42, on the other hand, is specified using predefined function blocks of the ALTERA MAX+PLUS II programming tool. The AV42 model could then be constructed using MODCHK. Both logics contain internal memories and delays, which increases the model state space.

Finally, VTT also verified the joint operation of PS and PACS, by creating encapsulated MODCHK function blocks out of the VHDL code submodules of the PC11, and then constructing a joint model of PS, AV42, and PC11 application logic. Having all the related logics in one model enabled VTT to verify how PS and PACS operate together to perform, e.g., periodic tests.

The verified properties were formalized in temporal logic by VTT experts, based on natural-language functional requirements stated in Areva's requirement specification documents for the PS, and functional descriptions in the user manuals for the PACS modules.

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."

3.2 Loviisa NPP I&C Renewal Project (ELSA)

Operated by the utility Fortum, the Loviisa NPP includes two pressurized water reactors of the type VVER-440. Loviisa 1 started operation in 1977, and Loviisa 2 in 1980. In the early 2000s, an I&C renewal project called LARA was started, but eventually discontinued in 2014 due to delays in project implementation. A new project called ELSA is now underway, with Rolls-Royce delivering the required I&C systems, based on Spinline technology.

Since the LARA project, VTT has performed independent, third-party review of the new I&C systems, on commission from Fortum [13]. In ELSA, 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), and the functional design of the Manual Backup System (MBS). Planned future work on the Preventive Protection System (PPS) will involve analyzing how the new Spinline based systems operate together with already installed systems developed by another supplier. The evaluated systems belong to Finnish safety classes SC2 and SC3, with the exception of RPCS, which is a non-safety system.

In ELSA, VTT has used MODCHK to model the Spinline function block diagrams. The verified properties have also been formalized in temporal logic by VTT experts, based on Fortum's and Rolls-Royce's functional requirement specification documents, as well as documentation related to the old analogue I&C system implementation. Based on the results, design modifications have been made to ELSA systems (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.

The biggest practical challenge is that the requirements — even in the nuclear I&C domain — are not always well defined. Especially in renewal projects, the original requirements may not even exist in written form, but expressed in logic diagrams. Even if the requirements for the old analogue I&C are formally defined, they do not consider issues specific to digital technology (e.g., time cycles, signal validity, initial/default states). In order to utilize the full potential of formal verification, requirements should always be formally defined between the system supplier and the customer, before the contract is signed.

In order to verify the I&C application logic in an efficient way, and as early as possible — in parallel with system design — model checking needs to be integrated to the I&C system design tools. Currently, Fortum and VTT are forerunners in such tool development, with a common project to integrate MODCHK with the Apros [18] dynamic process simulator, enabling formal verification as an integral part of I&C system design. It is often argued that formal verification is time-consuming, and offers little benefit given the costs. With integrated tools, model checking of I&C systems based on function block diagrams can be performed in a few days, provided that the requirements are well defined.

3.3 Hanhikivi 1 NPP Functional Architecture

Hanhikivi 1 is a NPP planned to be built in Pyhäjoki, Finland. In June 2015, the utility Fennovoima submitted a construction license application for an AES-2006 type pressurized water reactor, to be supplied by RAOS Project Oy, a subsidiary of Rusatom Energy International.

RAOS Project Oy and its sub-suppliers prepare the upper-level documentation related to safety engineering using the Advanced Licensing and Safety Engineering Method ADLAS® [19], developed by

Fortum. ADLAS® is used to prepare unified and hierarchical licensing documentation to demonstrate the compliance of the plant design with the Finnish regulatory requirements. In particular, the *functional architecture* defines and describes the safety functions of Hanhikivi 1 in an early life-cycle stage. The functionality of the safety functions is presented using function block diagrams that are platform independent, but similar to the diagrams used by I&C vendors. In the later design stages to follow, the functional architecture is then used as input for discipline specific technical architecture design, including the I&C architecture. Later, the safety functions presented in the functional architecture are elaborated into clear and unambiguous input for the actual I&C system design.

High quality of the functional architecture is essential for successful safety I&C design, since the architecture actually defines the functionality of the safety functions to be implemented. Changes in the functionality of the safety functions during later life-cycle stages can cause significant delays for the project, because the changes would require changes in the I&C design, which would call for additional V&V.

On commission from Fennovoima, VTT has evaluated preliminary versions of the Hanhikivi 1 functional architecture. The evaluation focused on the safety functions that were considered more complex than others (due to the use of, e.g., memory or delay function blocks, or feedback loops). VTT experts modelled the function block diagrams using MODCHK, and formalized the verified properties based on natural language requirements also listed in the ADLAS® documentation.

VTT's work demonstrated that model checking is a feasible method in the assessment of functional architectures, and it can help improve the quality of high-level safety function design. Consequently, use of formal methods like model checking is planned to have a role in the assessment of the Hanhikivi 1 functional architecture. Fennovoima also expects that model checking will be used to assess the eventual detailed design of safety classified I&C functions, where applicable.

4 IDENTIFIED DESIGN ISSUES

VTT has collected data on the design issues identified in practical customer projects between the years 2008 and 2016. Here, a design issue means that model checking revealed a practical example of potential scenario, where the I&C application logic ends up in a state that is contrary to a stated functional requirement. The issues are listed in Table III. In addition to a short general description, some characteristics of the issues are shown (e.g., was it a spurious actuation scenario? Did it involve very exact timing, or human actions?).

The reader should note that the issues in Table III are about a single system not fulfilling a stated requirement in some scenario, regardless of how unlikely that scenario is. VTT wishes to emphasize that the safety relevance of the issues is not considered (and may be insignificant or purely theoretical). Each system model has also been considered in isolation, i.e., not accounting for the characteristics of the controlled process — or the environment in some other sense — which means that any issue might be practically irrelevant considering the proper context. The items are not listed in the order of their discovery.

A more detailed presentation of the data can be found in [5].

Table III. List of generalized design issues identified using model checking.

#	Generalized description	S	F	I	T	O
1	Short signal pulses interfere with a test logic and lead to actuation	S		I	T	O
2	Test signal stored in delay block leads to spurious actuation after test.	S		I		
3	Test signal stored in memory leads to spurious actuation after test.	S				
4	A reset signal is ignored, if it arrives at a specific time.	S			T	O
5	Due to maintenance activity, a signal remains set without a cause.	S		I		O
6	Due to maintenance activity, a signal is spuriously set.	S		I		O
7	Invalid data on startup, improper operator action lead to actuation.	S			T	O
8	Spurious actuation commands are given on system start up.	S				
9	Two consecutive commands lead to an overtly long actuation signal.	S			T	
10	Overlapping commands lead to an overtly long actuation signal.	S			T	
11	Conflicting actuation commands are sent.	S			T	
12	Invalid signals trigger conflicting commands.	S				
13	Conflicting inputs trigger conflicting commands.	S				
14	A functional requirement is incorrect.	S				
15	A safety command is given without valid actuation criteria.	S			T	
16	Uncharacteristic inputs lead to inhibition of safety commands.		F			
17	Configuration of delays means that a signal is blocked.		F			
18	Configuration of delays means that a signal is blocked.		F	I		
19	After fluctuating inputs, operator can perform a forbidden action.				T	O
20	Fluctuating input data leads to the incorrect operational state.				T	O
21	Conflicting internal variables are set on fluctuating input data.				T	
22	Redundant systems can be inhibited at the same time.			I	T	
23	A lower priority signal overrules a higher priority one.					
24	Due to maintenance activity, a safety function is inhibited.			I	T	O
25	Due to maintenance activity, an actuator is inhibited.			I	T	O
26	A lower priority signal overrules a higher priority one.					
27	If input signals are on during start up, a safety function is inhibited.					
28	Redundant systems can be inhibited at the same time.				T	O
29	A test mode can only be deactivated via a wrong mechanism.					O
30	On conflicting input data, no operational state is selected.			I	T	
31	Conflicting operational modes selected on fluctuating input data.				T	O
32	An alarm can be acknowledged before it is received.				T	O
33	At system start up, an actuation command is inhibited.					
34	A functional requirement on timing is technically incorrect.					
35	Fluctuating inputs lead to short actuation command bursts.					
36	A safety command is inhibited on system start up.					
37	Very rapidly fluctuating commands are sent to an actuator.					
38	Very rapidly fluctuating commands are sent to an actuator.					
39	A safety command is inhibited after a delay.					
40	Signal validity processing fails.					
		37%	7%	17%	42%	32%

S = Spurious actuation
F = The logic is permanently frozen to an unwanted state (requires, e.g., a system restart to resolve)
I = The interaction of several systems is needed to reproduce the scenario.
T = The scenario involves very exact timing of events.
O = The scenario involves human (operator or maintenance personnel) actions.

Let us look at an exemplar issue — more specifically, issue number 31 in Table III. Figure II shows a modified and abstracted version of the verified application logic, containing only the blocks that are necessary to reproduce the scenario (the actual system model consisted of 74 functions blocks and had an I/O number of 23). The logic here consists of two functions: one for selecting the operational mode *a*, and another for selecting the mode *b*. The operator can use the *set_a* or *set_b* command to select the associated mode, and the selection is stored into a flip-flop memory block. If mode *a* is selected, and the signal *c* then becomes true, the mode is automatically changed to *b*. Communication between these functions creates a feedback loop, which is why the processing order is specified using a cycle delay block to "break the loop".

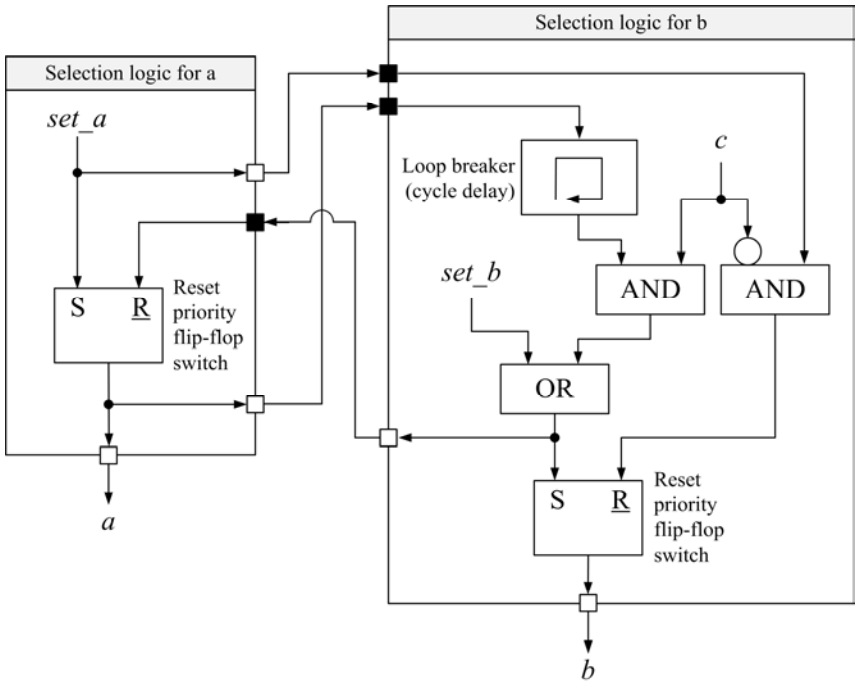

Figure II. Exemplar function block diagram — a mode selection logic

For the logic in Figure II, we can identify requirements such as:

- r_1: "The *set_a* command shall lead to *a*, if *b* is not already selected", or as a formal LTL property p_1: $\mathbf{G}((set\_a \wedge \neg\, b) \rightarrow a)$.
- r_2: "The *set_a* command shall reset *b*, if *c* is not active", or p_2: $\mathbf{G}((set\_a \wedge \neg\, c) \rightarrow \neg\, b)$.
- r_3: "The *set_b* command shall lead to *b* and reset *a*, if *set_a* is not active", or p_3: $\mathbf{G}((set\_b \wedge \neg\, set\_a) \rightarrow (b \wedge \neg\, a))$.
- r_4: "If *a* has been selected, the signal *c* shall change the selection to *b*", or p_4: $\mathbf{G}((\mathbf{Y}a \wedge c) \rightarrow b)$.
- r_5: "Only one mode (*a* or *b*) shall be active at the same time", or p_5: $\mathbf{G}\neg(a \wedge b)$

The **Y** operator in p_4 is needed to account for the cycle delay element.

We can also specify properties to address spurious actuation [5], such as $\mathbf{G}(a \rightarrow \mathbf{O}\ set\_a)$ and $\mathbf{G}(b \rightarrow \mathbf{O}\ (set\_b \vee (c \wedge \mathbf{Y}a)))$. Notably, p_5 also states that something should *not* happen.

Verification results indicate that p_5 is false. The model checked returns a counterexample which is visualized in Figure III.

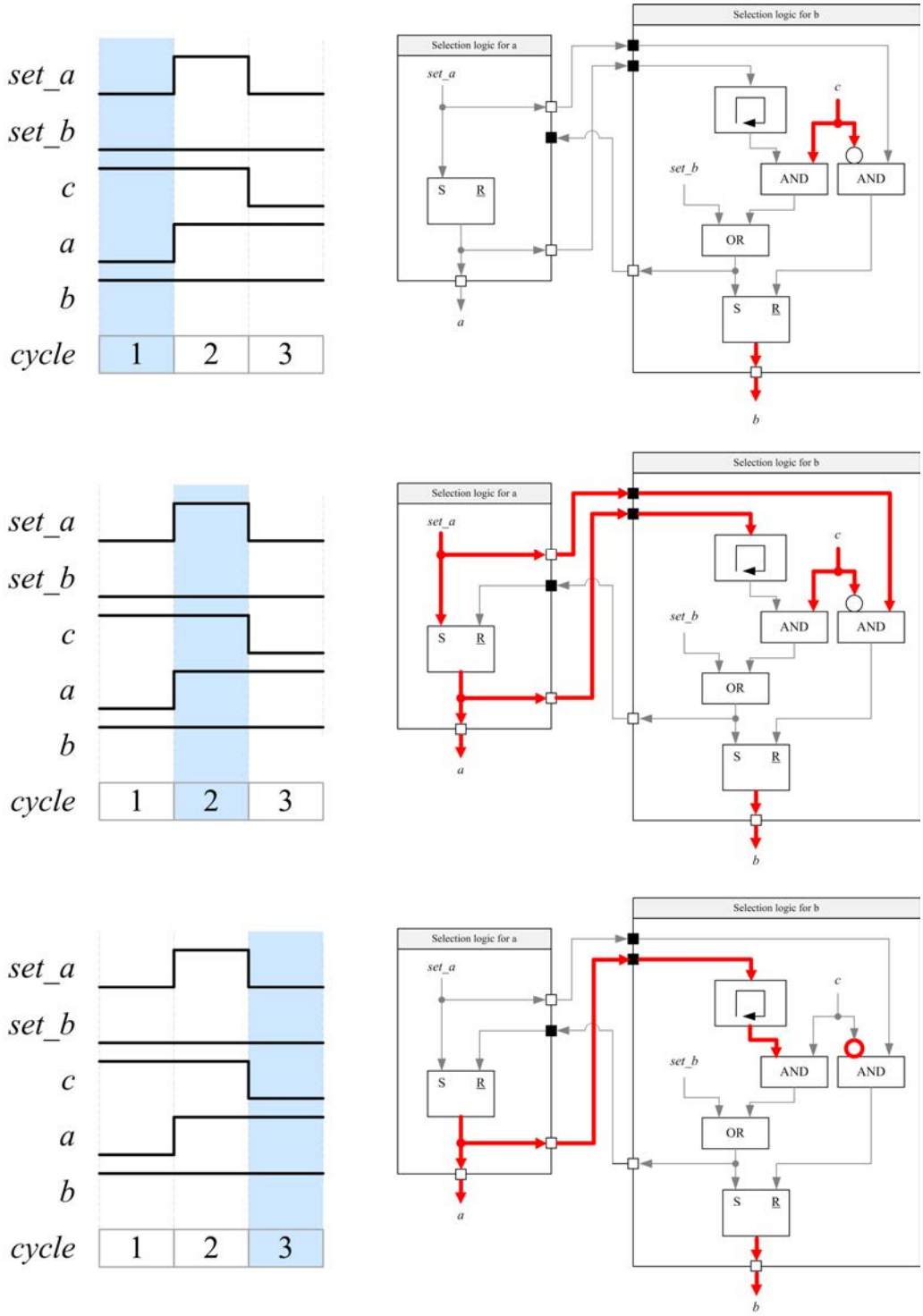

Figure III. A counterexample where both modes *a* and *b* are set at the same time

In the counterexample scenario, the mode *b* is first selected, and the signal *c* is also on. The operator then activates the *set_a* command, which does not reset *b*, because *c* is still on. At the same time, *c* does not reset *a* because the "loop breaker" block is delaying the *a* signal. Finally, on the next processing cycle, both *set_a* and *c* are reset at the exact same time, and both *a* and *b* remain set.

The issue hinges upon three factors that might be difficult to account for in test plans: 1) the operator invokes *set_a* at a time when *c* is also true, which is not necessarily feasible, 2) for some reason (e.g., a device failure), the *set_a* signal pulse is *extremely* short, and 3) two signals are reset at the *exact* same time.

Another practical example can be found in [5].

5 CONCLUSIONS

In Finland, model checking could be considered a nuclear industry practice. Since 2008, model checking has been used as a practical verification method in every major activity related to digital I&C in Finnish NPPs — be it a new-build or a renewal project. VTT's experience has shown that model checking can reveal design issues in I&C application logic that are otherwise hard to detect. It is practically the only viable option to detect possible spurious actuation scenarios caused by unintended functionality of I&C application software (or FPGA logic), a topic that is getting more and more attention.

Outside Finland (and outside the nuclear domain) model checking does not seem to be a commonplace I&C industry practice. Reasons might include unfamiliarity with the subject, or the perceived cost being too high. The latter point is debatable, given the proven benefits. Some expertise on topics like formal property specification is still required, but with the right tools, the work effort in most cases is calculated in days per evaluated system, rather than weeks.

6 ACKNOWLEDGMENTS

VTT's research on model checking has been funded by the Finnish Research Programme on Nuclear Power Plant Safety 2015-2018 (SAFIR2018). The MODCHK tool has been developed by Teemu Mätäsniemi of VTT and Janne Kauttio, formerly of VTT. We thank Mika Johansson of STUK for his feedback and comments.

7 REFERENCES

1. Bel V, BfS, CNSC, CSN, ISTec, ONR, SSM, and STUK, "Licensing of safety critical software for nuclear regulators, Common position of international nuclear regulators and authorised technical support organisations", Revision 2015 (2015).
2. E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, The MIT Press, Cambridge, Massachusetts (1999).
3. IAEA, *Technical Challenges in the Application and Licensing of Digital Instrumentation and Control Systems in Nuclear Power Plants*, International Atomic Energy Agency, Vienna (2015).
4. IEC 61508-3 Ed. 2.0: Functional safety of electrical/electronic/programmable electronic safety related systems – Part 3: Software requirements (2010).
5. A. Pakonen and K. Björkman, "Model Checking as a Protective Method Against Spurious Actuation of Industrial Control Systems", *Proceedings of the 27th European Safety and Reliability Conference (ESREL 2017)*, Portorož, Slovenia, June 18-22, to appear (2017).
6. M. Benedetti and C. Cimatti, "Bounded Model Checking for Past LTL", *Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2003. Lecture Notes in Computer Science*, **Volume 2619**, pp. 18-33 (2003).
7. 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).

8. A. Wassyng, M. Lawford M., “Lessons Learned from a Successful Implementation of Formal Methods in an Industrial Project”, In: K. Araki, S. Gnesi, D. Mandrioli (eds): *Formal Methods. FME 2003. Lecture Notes in Computer Science*, **Volume 2805**, pp. 133-153 (2003).
9. 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).
10. E. Németh, T. Bartha, “Formal Verification of Safety Functions by Reinterpretation of Functional Block Based Specifications”, In: D. Cofer, A. Fantechi A. (eds): *Formal Methods for Industrial Critical Systems. FMICS 2008. Lecture Notes in Computer Science*, **Volume 5596**, pp. 199-214 (2009).
11. B. F. Adiego, D. Darvas, E. B. Viñuela, J. C. Tournier, S. Bliudze, J. O. Blech, and V. M. G. Suárez, “Applying model checking to industrial-sized PLC programs”, *IEEE Transactions on Industrial Informatics*, **Volume 11**, pp. 1400-1410 (2015).
12. A. Wakankar, R. Mitra, A. K. Bhattacharjee, S. V. Shrikhande, S. D. Dhodapkar, B. B. Biswas, and R. K. Patil, “Formal model based methodology for developing software for nuclear applications”, *BARC Newsletter*, **Volume 318**, pp. 52-62 (2011).
13. A. Pakonen, J. Valkonen, S. Matinaho, and M. Hartikainen, ”Model Checking for Licensing Support in the Finnish Nuclear Industry”, *Proceedings of the International Symposium on Future I&C for Nuclear Power Plants (ISOFIC 2014)*, Jeju, Korea, Aug 24-28, pp. 1-9 (2014).
14. A. Pakonen, T. Mätäsniemi, J. Lahtinen, and T. Karhela, ”A Toolset for Model Checking of PLC Software”, *Proceedings of the 18th IEEE Conference on Emerging Technologies and Factory Automation (ETFA2013)*, Cagliari, Italy, Sep 10-13, pp. 1-8 (2013).
15. T. Ovatman, A. Aral, D. Polat, and A. O. Ünver, "An overview of model checking practices on verification of PLC software", *Software & Systems Modeling*, **Volume 15**, pp. 937-960 (2016).
16. A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella, “NuSMV 2: An Open Source Tool for Symbolic Model Checking”, *Proceedings of the 14th International Conference on Computer-Aided Verification (CAV 2002)*, Copenhagen, Denmark, July 27-31, p. 359-364 (2002).
17. A. Pakonen, “Verification of FPGA application design by model checking”, http://www.vtt.fi/inf/julkaisut/muut/2016/OA-Verification-of-FPGA-Application%20.pdf (2016).
18. Apros — Process simulator software for nuclear and thermal power plant applications, http://www.apros.fi/en/.
19. P. Nuutinen, S. Sipola and A. Rantakaulio, “Advanced Licensing and Safety Engineering Method - ADLAS®”, *Nuclear Science and Technology Symposium (NST2016)*, Helsinki, Finland, Nov 2-3 (2016).