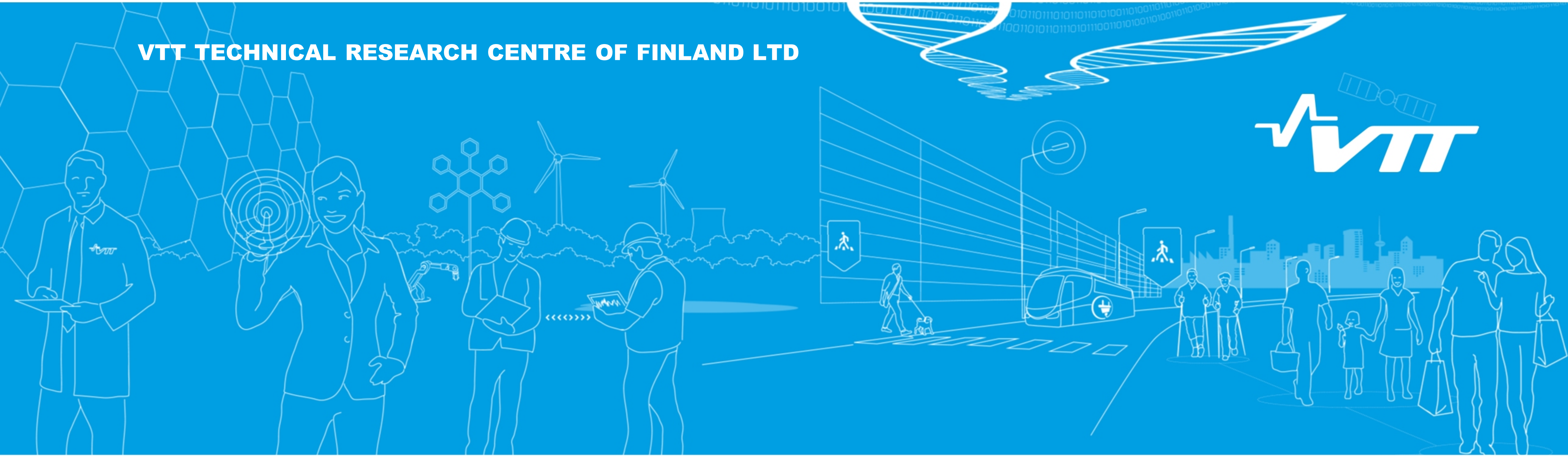
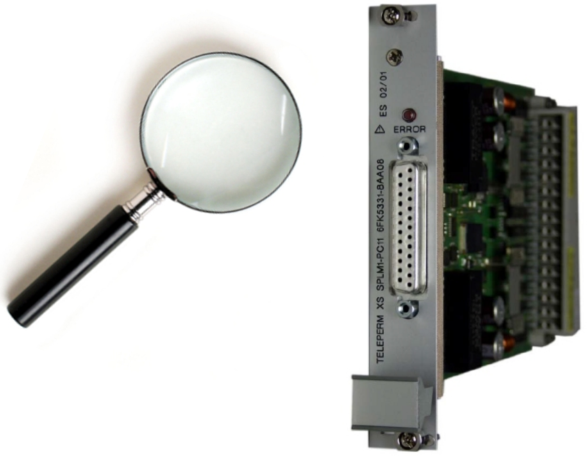
Title	Verification of FPGA application design by model checking
Author(s)	Pakonen, Antti
Citation	9th International Workshop on the Application of Field Programmable Gate Arrays in Nuclear Power Plants, 3 - 6 October 2016, Lyon, France
Date	2016
Rights	This article may be downloaded for personal use only.

VTT TECHNICAL RESEARCH CENTRE OF FINLAND LTD





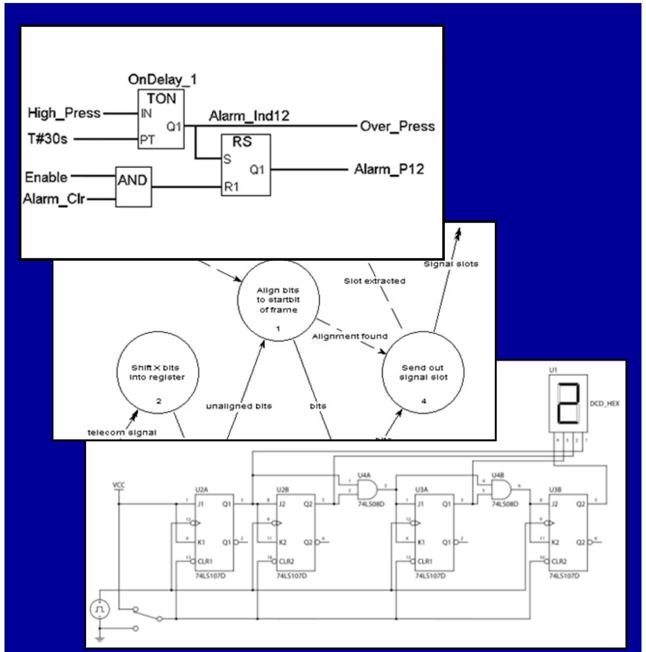
Verification of FPGA application design by model checking

9th Workshop on the application of FPGAs in NPPs

Antti Pakonen

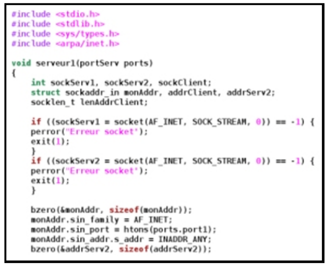
VTT Technical Research Centre of Finland Ltd

Scope: Verification of application design

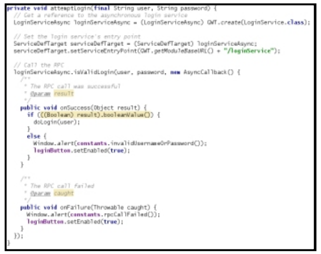


Program / Logic

Function Block Diagram
FPGA schematic
VHDL



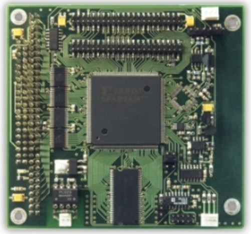
```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <arpa/inet.h>

void serveur1(portServ ports)
{
    int sockServ1, sockServ2, sockClient;
    struct sockaddr_in monAddr, addrClient, addrServ2;
    socklen_t lenAddrClient;

    if ((sockServ1 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Erreur socket");
    exit(1);
    }
    if ((sockServ2 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Erreur socket");
    exit(1);
    }

    bzero(&monAddr, sizeof(monAddr));
    monAddr.sin_family = AF_INET;
    monAddr.sin_port = htons(ports.port1);
    monAddr.sin_addr.s_addr = INADDR_ANY;
    bzero(&addrServ2, sizeof(addrServ2));
```



```
private void attemptLogin(final String user, String password) {
  // Get a reference to the asynchronous login service
  LoginServiceAsync loginServiceAsync = (LoginServiceAsync) GWT.create(LoginService.class);

  // Set the login service's entry point
  ServiceDefTarget serviceDefTarget = (ServiceDefTarget) loginServiceAsync;
  serviceDefTarget.setServiceEntryPoint(GWT.getModuleBaseURL() + "/loginService");

  // Call the RPC
  loginServiceAsync.isValidLogin(user, password, new AsyncCallback() {
    /**
     * The RPC call was successful
     * @param result
     */
    public void onSuccess(Object result) {
      if (((Boolean) result).booleanValue()) {
        doLogin(user);
      }
      else {
        Window.alert(constants.invalidUsernameOrPassword());
        loginButton.setEnabled(true);
      }
    }

    /**
     * The RPC call failed
     * @param caught
     */
    public void onFailure(Throwable caught) {
      Window.alert(constants.rpcCallFailed());
      loginButton.setEnabled(true);
    }
  });
}
```

Code / Netlist

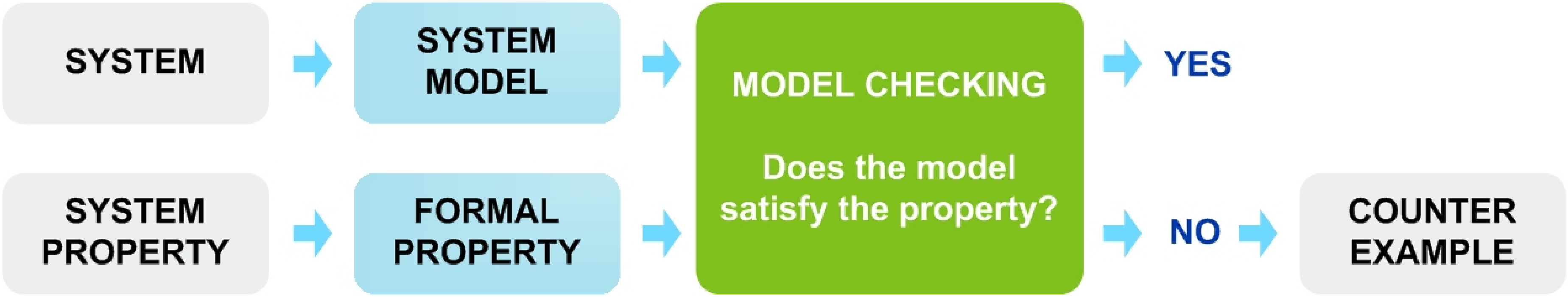
C, Java…
Assembly
FPGA netlist



Embedded System

PLC, FPGA,
PC, µC…

Model checking


SYSTEM

SYSTEM MODEL

SYSTEM PROPERTY

FORMAL PROPERTY

MODEL CHECKING

Does the model satisfy the property?

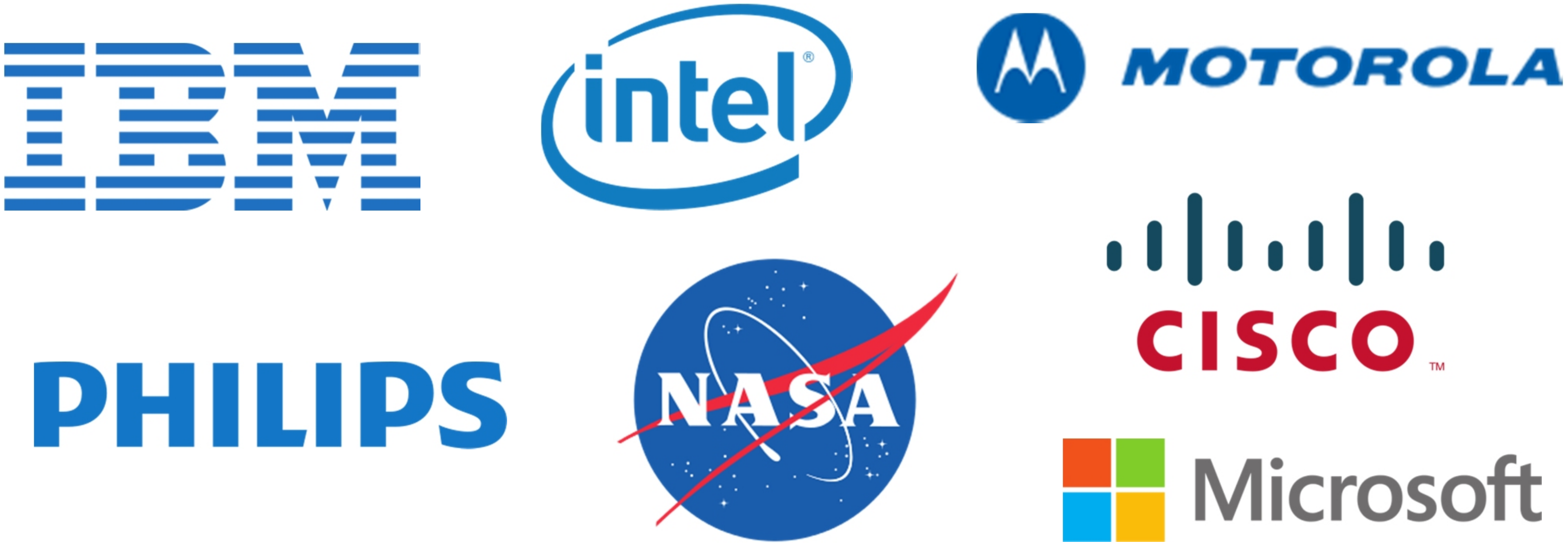
YES

NO

COUNTER EXAMPLE

A short history

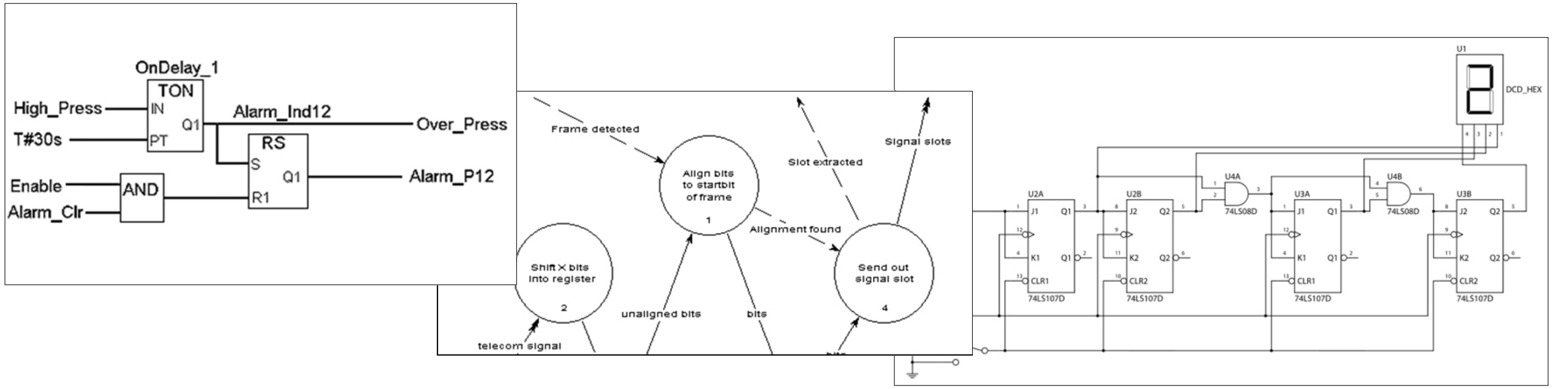
- Theory developed in early 1980s
- 1990s: hardware verification
- 2000s: software verification



Model checking of control logic design

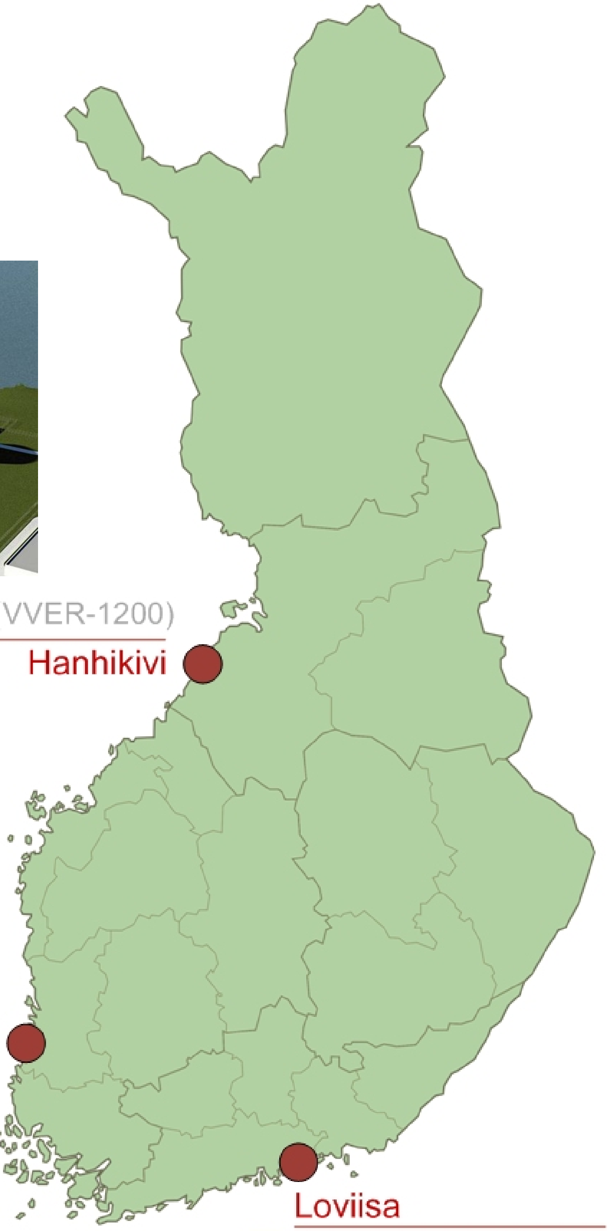
- Model checking **does** not apply to the evaluation of sophisticated control loops…
- …but it is **very efficient** in the context of safety critical systems!

- Faults can be found in systems that have already undergone traditional V&V.
- Faults often involve scenarios that are difficult to come up with.



NPPs in Finland

Image: Fennovoima



⬜ FH1 (VVER-1200)

Hanhikivi

🟨 **OL1** (BWR) **1979**

🟨 **OL2** (BWR) **1983**

⬜ OL3 (EPR)

Olkiluoto

Loviisa

🟨 **LO1** (VVER-440) **1977**

🟨 **LO2** (VVER-440) **1980**

Image: Hannu Tuovila / TVO

VTT customer projects

STUK

Olkiluoto 3 (under construction)

- Evaluation of NPP I&C system designs 2008-2011
- Evaluation of Olkiluoto 3 Protection System 2015
- **Evaluation of Olkiluoto 3 PACS 2015**

Fortum

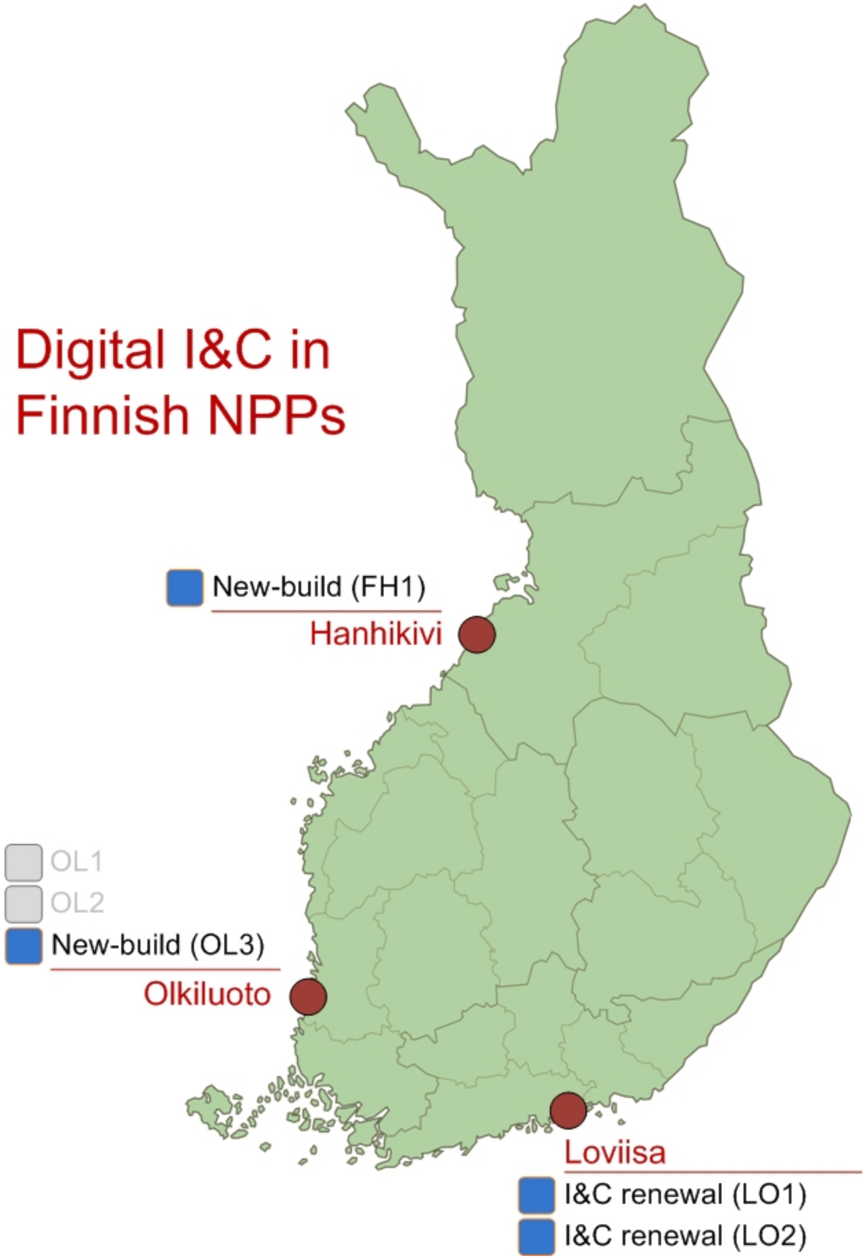
Loviisa 1 & 2 I&C modernization

- Verification of nuclear automation 2009
- Verification of nuclear I&C in the LARA project 2012-2014
- Verification of nuclear I&C in the ELSA project 2016

FENNOVOIMA

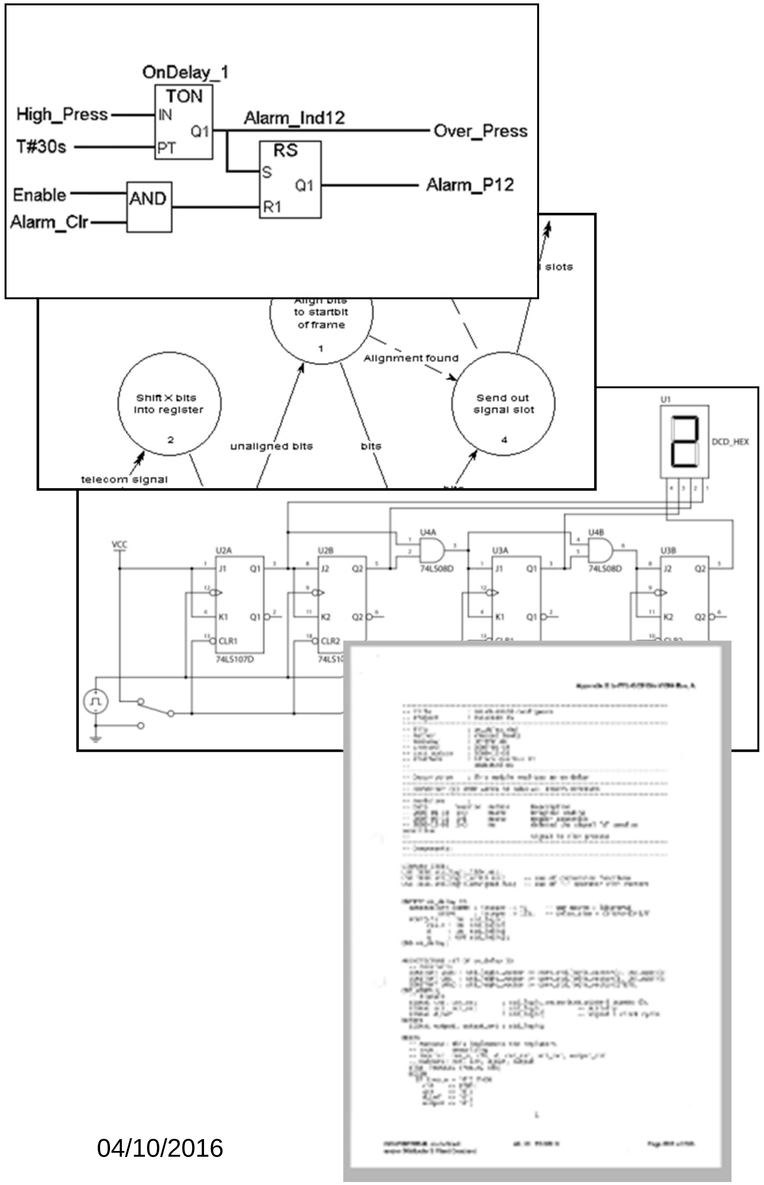
Hanhikivi 1 (decision-in-principle)

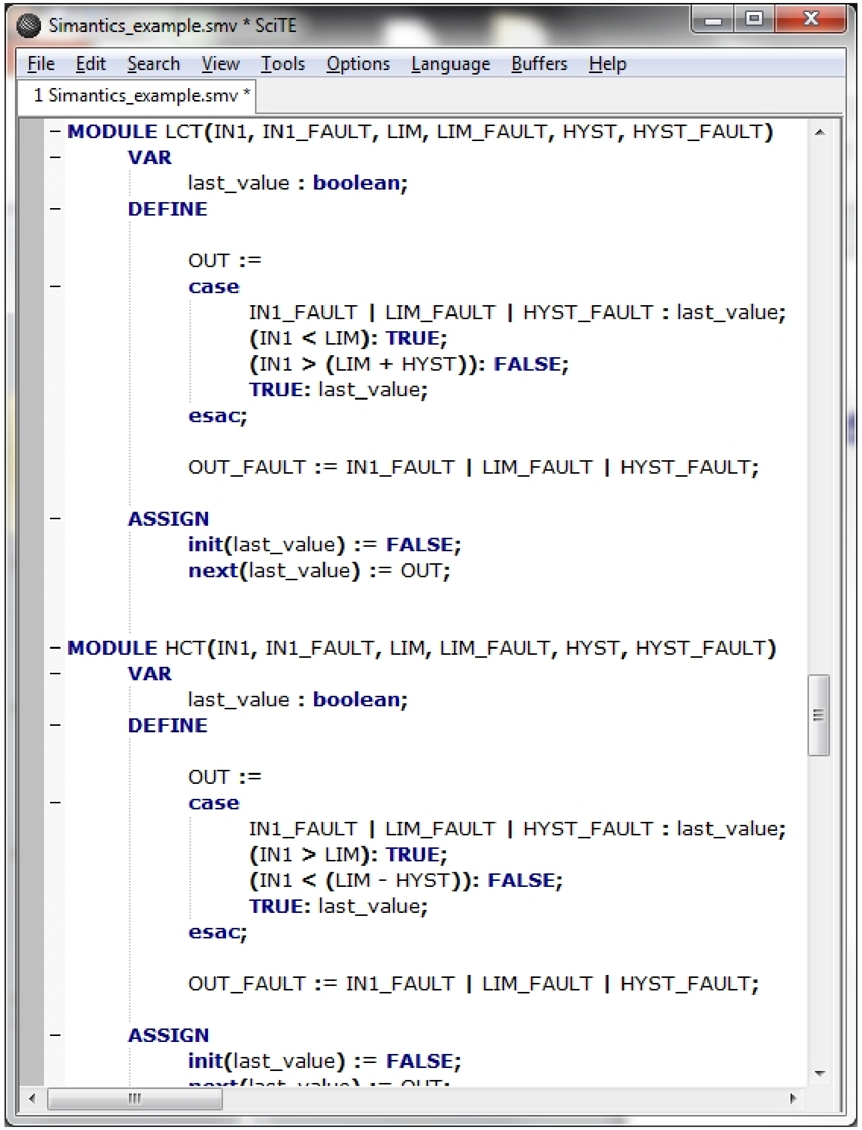
- Model checking of functional, architecture-level I&C 2016



Digital I&C in Finnish NPPs

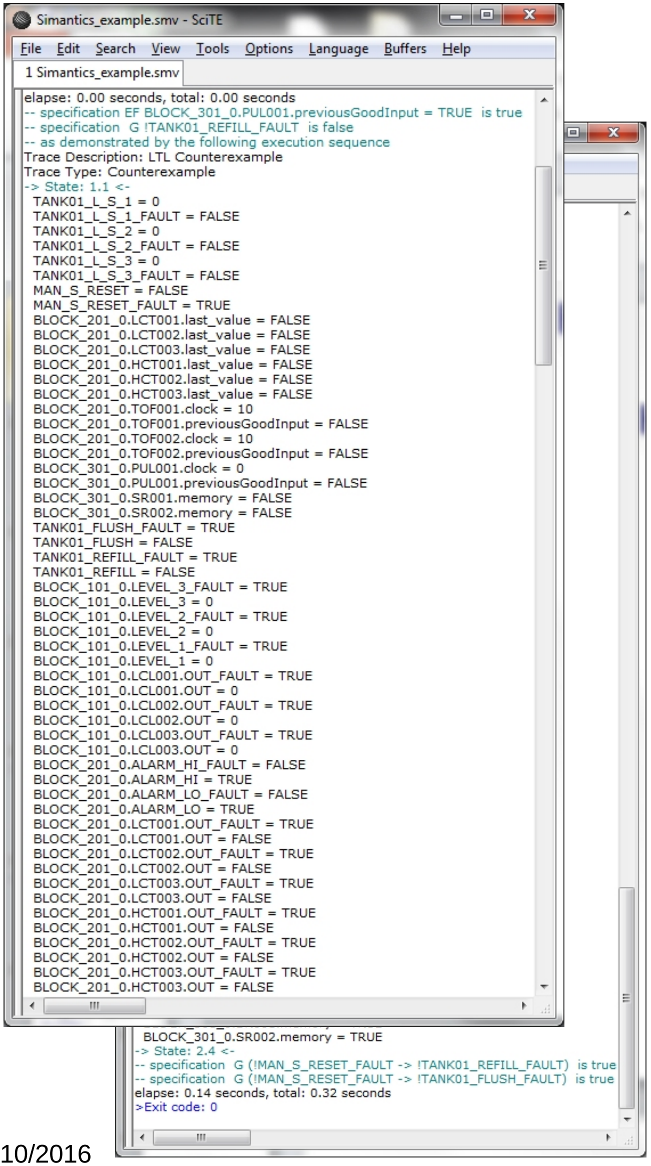
Challenges: Modelling




Simantics_example.smv * SciTE
File Edit Search View Tools Options Language Buffers Help
1 Simantics_example.smv *

```
MODULE LCT(IN1, IN1_FAULT, LIM, LIM_FAULT, HYST, HYST_FAULT)
        VAR
                last_value : boolean;
        DEFINE

                OUT :=
                case
                        IN1_FAULT | LIM_FAULT | HYST_FAULT : last_value;
                        (IN1 < LIM): TRUE;
                        (IN1 > (LIM + HYST)): FALSE;
                        TRUE: last_value;
                esac;

                OUT_FAULT := IN1_FAULT | LIM_FAULT | HYST_FAULT;

        ASSIGN
                init(last_value) := FALSE;
                next(last_value) := OUT;


MODULE HCT(IN1, IN1_FAULT, LIM, LIM_FAULT, HYST, HYST_FAULT)
        VAR
                last_value : boolean;
        DEFINE

                OUT :=
                case
                        IN1_FAULT | LIM_FAULT | HYST_FAULT : last_value;
                        (IN1 > LIM): TRUE;
                        (IN1 < (LIM - HYST)): FALSE;
                        TRUE: last_value;
                esac;

                OUT_FAULT := IN1_FAULT | LIM_FAULT | HYST_FAULT;

        ASSIGN
                init(last_value) := FALSE;
                next(last_value) := OUT;
```

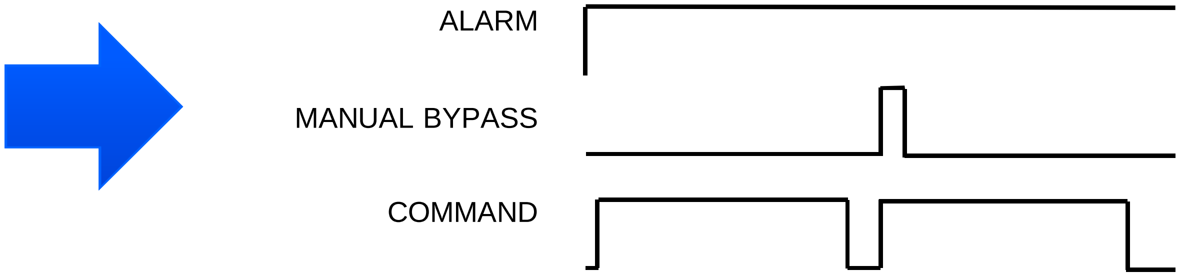
Challenges: Counterexample visualization



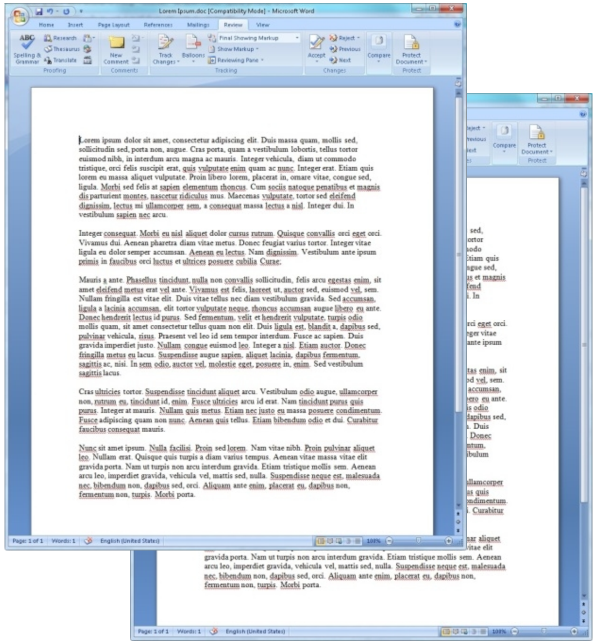
Simantics_example.smv - SciTE
File Edit Search View Tools Options Language Buffers Help
1 Simantics_example.smv

```
elapse: 0.00 seconds, total: 0.00 seconds
-- specification EF BLOCK_301_0.PUL001.previousGoodInput = TRUE  is true
-- specification  G !TANK01_REFILL_FAULT  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  TANK01_L_S_1 = 0
  TANK01_L_S_1_FAULT = FALSE
  TANK01_L_S_2 = 0
  TANK01_L_S_2_FAULT = FALSE
  TANK01_L_S_3 = 0
  TANK01_L_S_3_FAULT = FALSE
  MAN_S_RESET = FALSE
  MAN_S_RESET_FAULT = TRUE
  BLOCK_201_0.LCT001.last_value = FALSE
  BLOCK_201_0.LCT002.last_value = FALSE
  BLOCK_201_0.LCT003.last_value = FALSE
  BLOCK_201_0.HCT001.last_value = FALSE
  BLOCK_201_0.HCT002.last_value = FALSE
  BLOCK_201_0.HCT003.last_value = FALSE
  BLOCK_201_0.TOF001.clock = 10
  BLOCK_201_0.TOF001.previousGoodInput = FALSE
  BLOCK_201_0.TOF002.clock = 10
  BLOCK_201_0.TOF002.previousGoodInput = FALSE
  BLOCK_301_0.PUL001.clock = 0
  BLOCK_301_0.PUL001.previousGoodInput = FALSE
  BLOCK_301_0.SR001.memory = FALSE
  BLOCK_301_0.SR002.memory = FALSE
  TANK01_FLUSH_FAULT = TRUE
  TANK01_FLUSH = FALSE
  TANK01_REFILL_FAULT = TRUE
  TANK01_REFILL = FALSE
  BLOCK_101_0.LEVEL_3_FAULT = TRUE
  BLOCK_101_0.LEVEL_3 = 0
  BLOCK_101_0.LEVEL_2_FAULT = TRUE
  BLOCK_101_0.LEVEL_2 = 0
  BLOCK_101_0.LEVEL_1_FAULT = TRUE
  BLOCK_101_0.LEVEL_1 = 0
  BLOCK_101_0.LCL001.OUT_FAULT = TRUE
  BLOCK_101_0.LCL001.OUT = 0
  BLOCK_101_0.LCL002.OUT_FAULT = TRUE
  BLOCK_101_0.LCL002.OUT = 0
  BLOCK_101_0.LCL003.OUT_FAULT = TRUE
  BLOCK_101_0.LCL003.OUT = 0
  BLOCK_201_0.ALARM_HI_FAULT = FALSE
  BLOCK_201_0.ALARM_HI = TRUE
  BLOCK_201_0.ALARM_LO_FAULT = FALSE
  BLOCK_201_0.ALARM_LO = TRUE
  BLOCK_201_0.LCT001.OUT_FAULT = TRUE
  BLOCK_201_0.LCT001.OUT = FALSE
  BLOCK_201_0.LCT002.OUT_FAULT = TRUE
  BLOCK_201_0.LCT002.OUT = FALSE
  BLOCK_201_0.LCT003.OUT_FAULT = TRUE
  BLOCK_201_0.LCT003.OUT = FALSE
  BLOCK_201_0.HCT001.OUT_FAULT = TRUE
  BLOCK_201_0.HCT001.OUT = FALSE
  BLOCK_201_0.HCT002.OUT_FAULT = TRUE
  BLOCK_201_0.HCT002.OUT = FALSE
  BLOCK_201_0.HCT003.OUT_FAULT = TRUE
  BLOCK_201_0.HCT003.OUT = FALSE
```

```
  BLOCK_301_0.SR002.memory = TRUE
-> State: 2.4 <-
-- specification  G (!MAN_S_RESET_FAULT -> !TANK01_REFILL_FAULT)  is true
-- specification  G (!MAN_S_RESET_FAULT -> !TANK01_FLUSH_FAULT)  is true
elapse: 0.14 seconds, total: 0.32 seconds
>Exit code: 0
```



Challenges: Requirement formalization



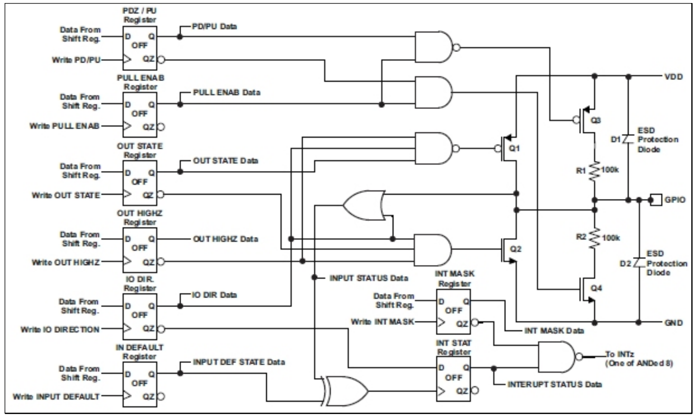
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis massa quam, mollis sed, sollicitudin sed, porta non, augue. Cras porta, quam a vestibulum lobortis, tellus tortor euismod nibh, in interdum arcu magna ac mauris. Integer vehicula, diam ut commodo tristique, orci felis suscipit erat, quis vulputate enim quam ac nunc. Integer erat. Etiam quis lorem eu massa aliquet vulputate. Proin libero lorem, placerat in, ornare vitae, congue sed, ligula. Morbi sed felis at sapien elementum rhoncus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas vulputate, tortor sed eleifend dignissim, lectus mi ullamcorper sem, a consequat massa lectus a nisl. Integer dui. In vestibulum sapien nec arcu.

Integer consequat. Morbi eu nisl aliquet dolor cursus rutrum. Quisque convallis orci eget orci. Vivamus dui. Aenean pharetra diam vitae metus. Donec feugiat varius tortor. Integer vitae ligula eu dolor semper accumsan. Aenean eu lectus. Nam dignissim. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Mauris a ante. Phasellus tincidunt, nulla non convallis sollicitudin, felis arcu egestas enim, sit amet eleifend metus erat vel ante. Vivamus est felis, laoreet ut, auctor sed, euismod vel, sem. Nullam fringilla est vitae elit. Duis vitae tellus nec diam vestibulum gravida. Sed accumsan, ligula a lacinia accumsan, elit tortor vulputate neque, rhoncus accumsan augue libero eu ante. Donec hendrerit lectus id purus. Sed fermentum, velit et hendrerit vulputate, turpis odio mollis quam, sit amet consectetur tellus quam non elit. Duis ligula est, blandit a, dapibus sed, pulvinar vehicula, risus. Praesent vel leo id sem tempor interdum. Fusce ac sapien. Duis gravida imperdiet justo. Nullam congue euismod leo. Integer a nisl. Etiam auctor. Donec fringilla metus eu lacus. Suspendisse augue sapien, aliquet lacinia, dapibus fermentum, sagittis ac, nisi. In sem odio, auctor vel, molestie eget, posuere in, enim. Sed vestibulum sagittis lacus.

Cras ultricies tortor. Suspendisse tincidunt aliquet arcu. Vestibulum odio augue, ullamcorper non, rutrum eu, tincidunt id, enim. Fusce ultricies arcu id erat. Nam tincidunt purus quis purus. Integer at mauris. Nullam quis metus. Etiam nec justo eu massa posuere condimentum. Fusce adipiscing quam non nunc. Aenean quis tellus. Etiam bibendum odio et dui. Curabitur faucibus consequat mauris.

Nunc sit amet ipsum. Nulla facilisi. Proin sed lorem. Nam vitae nibh. Proin pulvinar aliquet leo. Nullam erat. Quisque quis turpis a diam varius tempus. Aenean vitae massa vitae elit gravida porta. Nam ut turpis non arcu interdum gravida. Etiam tristique mollis sem. Aenean arcu leo, imperdiet gravida, vehicula vel, mattis sed, nulla. Suspendisse neque est, malesuada nec, bibendum non, dapibus sed, orci. Aliquam ante enim, placerat eu, dapibus non, fermentum non, turpis. Morbi porta.



(G !Alarm) | (!Alarm U (Alarm & F Shutdown -> (!Shutdown U ((Temperature < 55) & !Shutdown & X (!Shutdown U Feedback_pump = OFF))))

G ((Shutdown & ! ((T4_Level_M < 230))) -> (G (V15_Open) | ! V15_Open U (T4_Level_M < 230))))

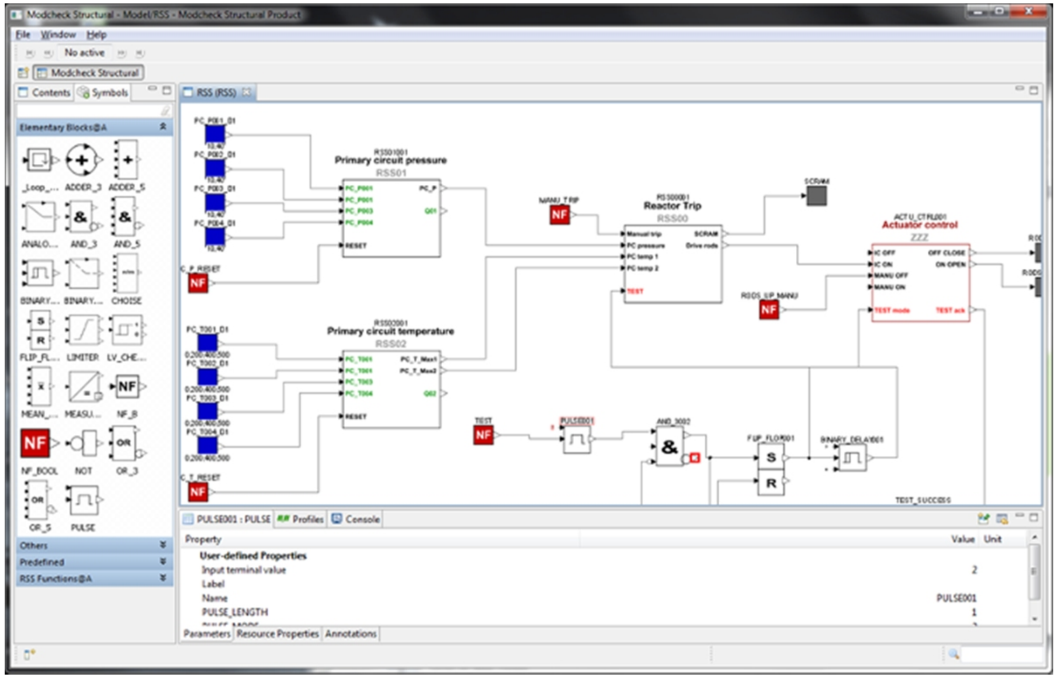
MODCHK – graphical tools for I&C verification [1]

- Vendor-specific, proprietary function block libraries [2]

- Structural, composite models

- Verification with NuSMV 2.6.0

- Counterexample animation



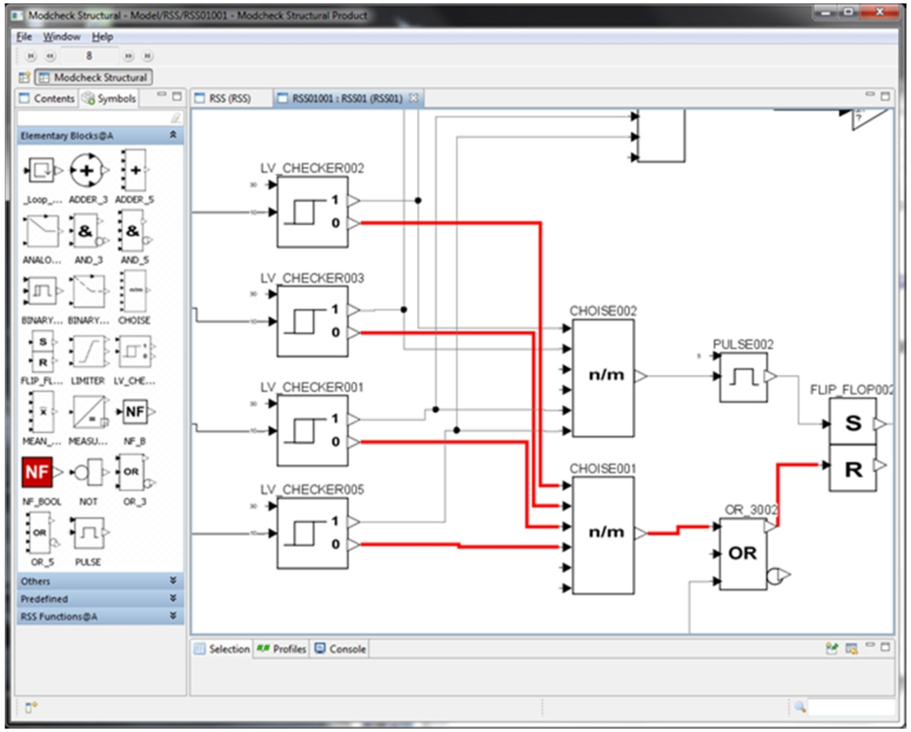
Property	Value	Unit
User-defined Properties		
Input terminal value	2	
Label		
Name	PULSE001	
PULSE_LENGTH	1	

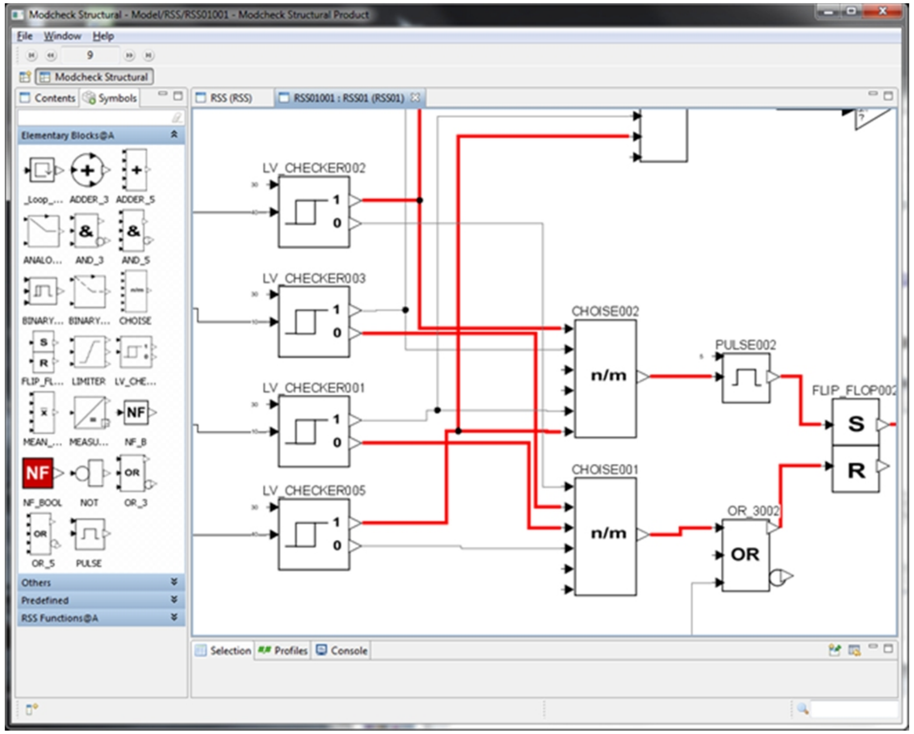
[1] https://www.simulationstore.com/node/52

[2] A. Pakonen, T. Mätäsniemi, J. Lahtinen, T. Karhela, A Toolset for Model Checking of PLC Software, 18th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA2013, 10-13 September 2013, Cagliari, Italy, Proceedings. IEEE (2013)

MODCHK: Counterexample visualization

- 2D animation, with numerical monitors for analogue signals





VHDL model checking

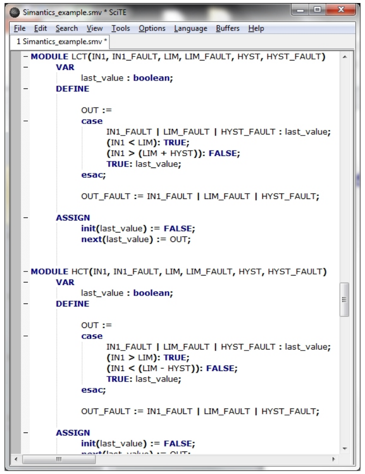
- **VHDL model checking tools** have been proposed. [1]
- ”Tweaking” (simplification / abstraction) is still needed in the case of, e.g., timed delays.
  - Megahertz scale clock cycle vs. second-scale delays -> large integer variables & state space explosion [2]

- Theoretical problems for VHDL finite-state verification:
  - Increasingly delayed signal assignment nested in an infinite loop [1]
  - Non-halting recursive function with local variables [1]

[1] Déharbe, D., Shankar, S., Clarke, E.M.: Model Checking VHDL with CV. Proceedings of the Second International Conference on Formal Methods in Computer-Aided Design. FMCAD '98. 508-514. Springer-Verlag, London, UK, 1998

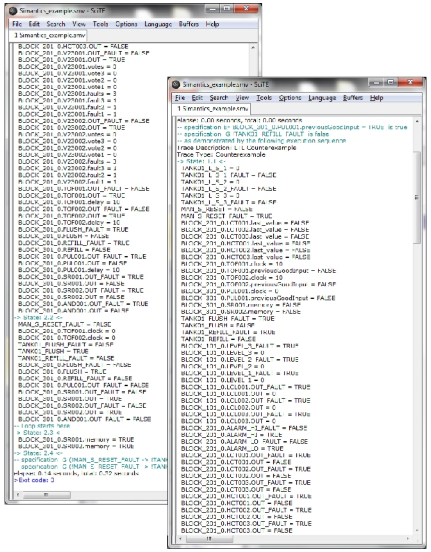
[2] Lahtinen, J., Ranta, J., Lötjönen L.: CORSICA 2013 work report: Test set generation, FPGA model checking, and fault injection. VTT Research Report VTT-R-00212-14, 2014.

VHDL model checking – the ”hard” way



Simantics_example.smv * SciTE
File Edit Search View Tools Options Language Buffers Help
1 Simantics_example.smv *

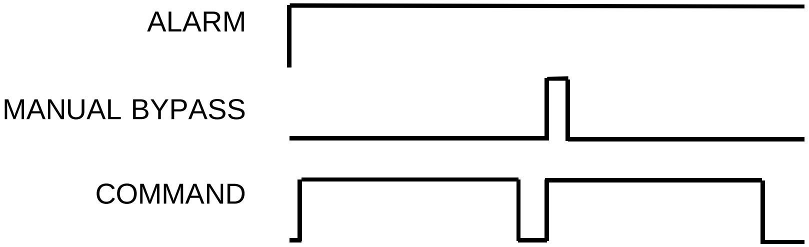
```
MODULE LCT(IN1, IN1_FAULT, LIM, LIM_FAULT, HYST, HYST_FAULT)
        VAR
                last_value : boolean;
        DEFINE

                OUT :=
                case
                        IN1_FAULT | LIM_FAULT | HYST_FAULT : last_value;
                        (IN1 < LIM): TRUE;
                        (IN1 > (LIM + HYST)): FALSE;
                        TRUE: last_value;
                esac;

                OUT_FAULT := IN1_FAULT | LIM_FAULT | HYST_FAULT;

        ASSIGN
                init(last_value) := FALSE;
                next(last_value) := OUT;


MODULE HCT(IN1, IN1_FAULT, LIM, LIM_FAULT, HYST, HYST_FAULT)
        VAR
                last_value : boolean;
        DEFINE

                OUT :=
                case
                        IN1_FAULT | LIM_FAULT | HYST_FAULT : last_value;
                        (IN1 > LIM): TRUE;
                        (IN1 < (LIM - HYST)): FALSE;
                        TRUE: last_value;
                esac;

                OUT_FAULT := IN1_FAULT | LIM_FAULT | HYST_FAULT;

        ASSIGN
                init(last_value) := FALSE;
```



Simantics_example.smv - SciTE
File Edit Search View Tools Options Language Buffers Help
1 Simantics_example.smv

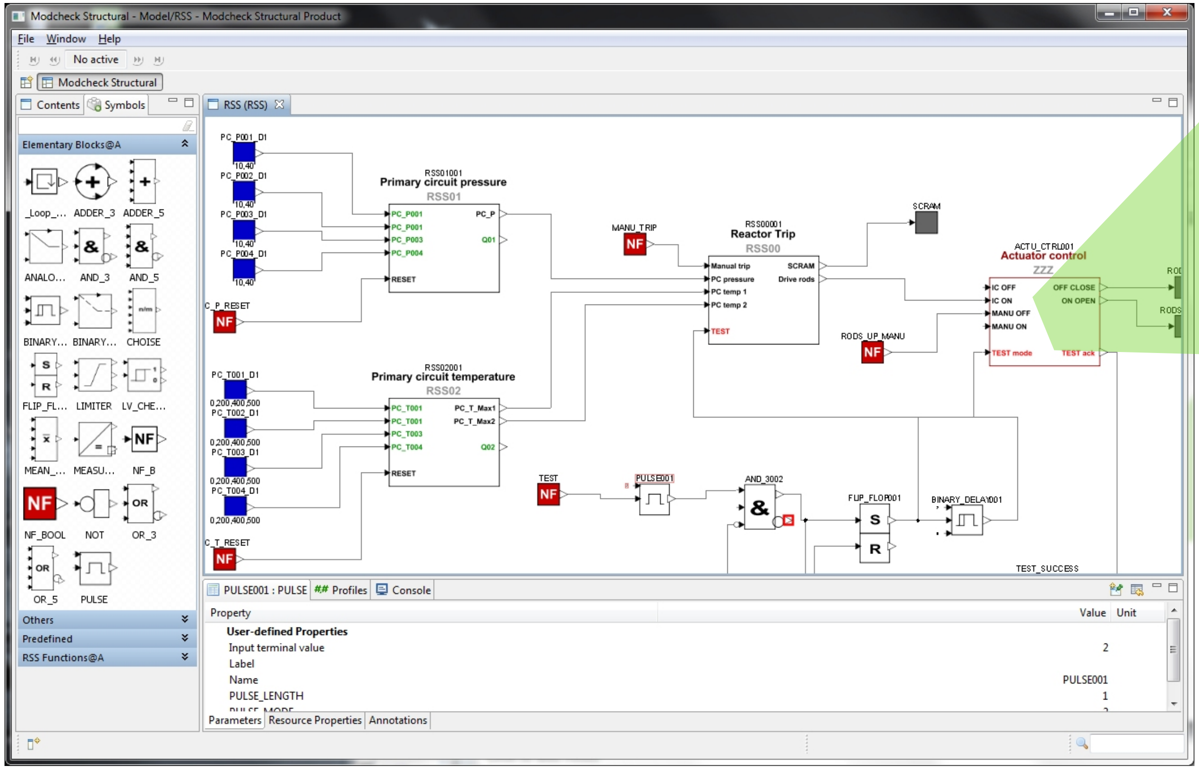
```
  BLOCK_201_0.HCT003.OUT = FALSE
  BLOCK_201_0.V23001.OUT_FAULT = FALSE
  BLOCK_201_0.V23001.OUT = TRUE
  BLOCK_201_0.V23001.votes = 0
  BLOCK_201_0.V23001.vote3 = 0
  BLOCK_201_0.V23001.vote2 = 0
  BLOCK_201_0.V23001.vote1 = 0
  BLOCK_201_0.V23001.faults = 3
  BLOCK_201_0.V23001.fault3 = 1
  BLOCK_201_0.V23001.fault2 = 1
  BLOCK_201_0.V23001.fault1 = 1
  BLOCK_201_0.V23002.OUT_FAULT = FALSE
  BLOCK_201_0.V23002.OUT = TRUE
  BLOCK_201_0.V23002.votes = 0
  BLOCK_201_0.V23002.vote3 = 0
  BLOCK_201_0.V23002.vote2 = 0
  BLOCK_201_0.V23002.vote1 = 0
  BLOCK_201_0.V23002.faults = 3
  BLOCK_201_0.V23002.fault3 = 1
  BLOCK_201_0.V23002.fault2 = 1
  BLOCK_201_0.V23002.fault1 = 1
  BLOCK_201_0.TOF001.OUT_FAULT = FALSE
  BLOCK_201_0.TOF001.OUT = TRUE
  BLOCK_201_0.TOF001.delay = 10
  BLOCK_201_0.TOF002.OUT_FAULT = FALSE
  BLOCK_201_0.TOF002.OUT = TRUE
  BLOCK_201_0.TOF002.delay = 10
  BLOCK_301_0.FLUSH_FAULT = TRUE
  BLOCK_301_0.FLUSH = FALSE
  BLOCK_301_0.REFILL_FAULT = TRUE
  BLOCK_301_0.REFILL = FALSE
  BLOCK_301_0.PUL001.OUT_FAULT = TRUE
  BLOCK_301_0.PUL001.OUT = FALSE
  BLOCK_301_0.PUL001.delay = 10
  BLOCK_301_0.SR001.OUT_FAULT = TRUE
  BLOCK_301_0.SR001.OUT = FALSE
  BLOCK_301_0.SR002.OUT_FAULT = TRUE
  BLOCK_301_0.SR002.OUT = FALSE
  BLOCK_301_0.AND001.OUT_FAULT = TRUE
  BLOCK_301_0.AND001.OUT = FALSE
-> State: 2.2 <-
  MAN_S_RESET_FAULT = FALSE
  BLOCK_201_0.TOF001.clock = 0
  BLOCK_201_0.TOF002.clock = 0
  TANK01_FLUSH_FAULT = FALSE
  TANK01_FLUSH = TRUE
  TANK01_REFILL_FAULT = FALSE
  BLOCK_301_0.FLUSH_FAULT = FALSE
  BLOCK_301_0.FLUSH = TRUE
  BLOCK_301_0.REFILL_FAULT = FALSE
  BLOCK_301_0.PUL001.OUT_FAULT = FALSE
  BLOCK_301_0.SR001.OUT_FAULT = FALSE
  BLOCK_301_0.SR001.OUT = TRUE
  BLOCK_301_0.SR002.OUT_FAULT = FALSE
  BLOCK_301_0.SR002.OUT = TRUE
  BLOCK_301_0.AND001.OUT_FAULT = FALSE
-- Loop starts here
-> State: 2.3 <-
  BLOCK_301_0.SR001.memory = TRUE
  BLOCK_301_0.SR002.memory = TRUE
-> State: 2.4 <-
-- specification G (!MAN_S_RESET_FAULT -> !TANK
-- specification G (!MAN_S_RESET_FAULT -> !TANK
elapse: 0.14 seconds, total: 0.32 seconds
>Exit code: 0
```

```
elapse: 0.00 seconds, total: 0.00 seconds
-- specification EF BLOCK_301_0.PUL001.previousGoodInput = TRUE  is true
-- specification G !TANK01_REFILL_FAULT  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  TANK01_L_S_1 = 0
  TANK01_L_S_1_FAULT = FALSE
  TANK01_L_S_2 = 0
  TANK01_L_S_2_FAULT = FALSE
  TANK01_L_S_3 = 0
  TANK01_L_S_3_FAULT = FALSE
  MAN_S_RESET = FALSE
  MAN_S_RESET_FAULT = TRUE
  BLOCK_201_0.LCT001.last_value = FALSE
  BLOCK_201_0.LCT002.last_value = FALSE
  BLOCK_201_0.LCT003.last_value = FALSE
  BLOCK_201_0.HCT001.last_value = FALSE
  BLOCK_201_0.HCT002.last_value = FALSE
  BLOCK_201_0.HCT003.last_value = FALSE
  BLOCK_201_0.TOF001.clock = 10
  BLOCK_201_0.TOF001.previousGoodInput = FALSE
  BLOCK_201_0.TOF002.clock = 10
  BLOCK_201_0.TOF002.previousGoodInput = FALSE
  BLOCK_301_0.PUL001.clock = 0
  BLOCK_301_0.PUL001.previousGoodInput = FALSE
  BLOCK_301_0.SR001.memory = FALSE
  BLOCK_301_0.SR002.memory = FALSE
  TANK01_FLUSH_FAULT = TRUE
  TANK01_FLUSH = FALSE
  TANK01_REFILL_FAULT = TRUE
  TANK01_REFILL = FALSE
  BLOCK_101_0.LEVEL_3_FAULT = TRUE
  BLOCK_101_0.LEVEL_3 = 0
  BLOCK_101_0.LEVEL_2_FAULT = TRUE
  BLOCK_101_0.LEVEL_2 = 0
  BLOCK_101_0.LEVEL_1_FAULT = TRUE
  BLOCK_101_0.LEVEL_1 = 0
  BLOCK_101_0.LCL001.OUT_FAULT = TRUE
  BLOCK_101_0.LCL001.OUT = 0
  BLOCK_101_0.LCL002.OUT_FAULT = TRUE
  BLOCK_101_0.LCL002.OUT = 0
  BLOCK_101_0.LCL003.OUT_FAULT = TRUE
  BLOCK_101_0.LCL003.OUT = 0
  BLOCK_201_0.ALARM_HI_FAULT = FALSE
  BLOCK_201_0.ALARM_HI = TRUE
  BLOCK_201_0.ALARM_LO_FAULT = FALSE
  BLOCK_201_0.ALARM_LO = TRUE
  BLOCK_201_0.LCT001.OUT_FAULT = TRUE
  BLOCK_201_0.LCT001.OUT = FALSE
  BLOCK_201_0.LCT002.OUT_FAULT = TRUE
  BLOCK_201_0.LCT002.OUT = FALSE
  BLOCK_201_0.LCT003.OUT_FAULT = TRUE
  BLOCK_201_0.LCT003.OUT = FALSE
  BLOCK_201_0.HCT001.OUT_FAULT = TRUE
  BLOCK_201_0.HCT001.OUT = FALSE
  BLOCK_201_0.HCT002.OUT_FAULT = TRUE
  BLOCK_201_0.HCT002.OUT = FALSE
  BLOCK_201_0.HCT003.OUT_FAULT = TRUE
  BLOCK_201_0.HCT003.OUT = FALSE
```



VHDL model checking with MODCHK

- Encapsulation of VHDL code
- Joint verification of µp and FPGA systems



Property	Value	Unit
User-defined Properties		
Input terminal value	2	
Label		
Name	PULSE001	
PULSE_LENGTH	1	

Application logic model checking: µp vs. FPGA

µp model checking	**FPGA** model checking
FB: Automatic model generation not possible, if the basic blocks are proprietary. Abstraction is also often needed.	**FB**: Automatic model generation not possible, if the basic blocks are proprietary. Abstraction is also often needed. **VHDL**: Automatic model generation not entirely feasible, e.g., timed delays have to be scaled.
FB: Graphical tools can be used for modelling & block-level counterexample animation.	**FB / VHDL**: Graphical tools can be used for modelling & block-level counterexample animation.

VTT customer projects

STUK

Olkiluoto 3 (under construction)

- Evaluation of NPP I&C system designs 2008-2011
- Evaluation of Olkiluoto 3 Protection System 2015
- **Evaluation of Olkiluoto 3 PACS 2015**

Fortum

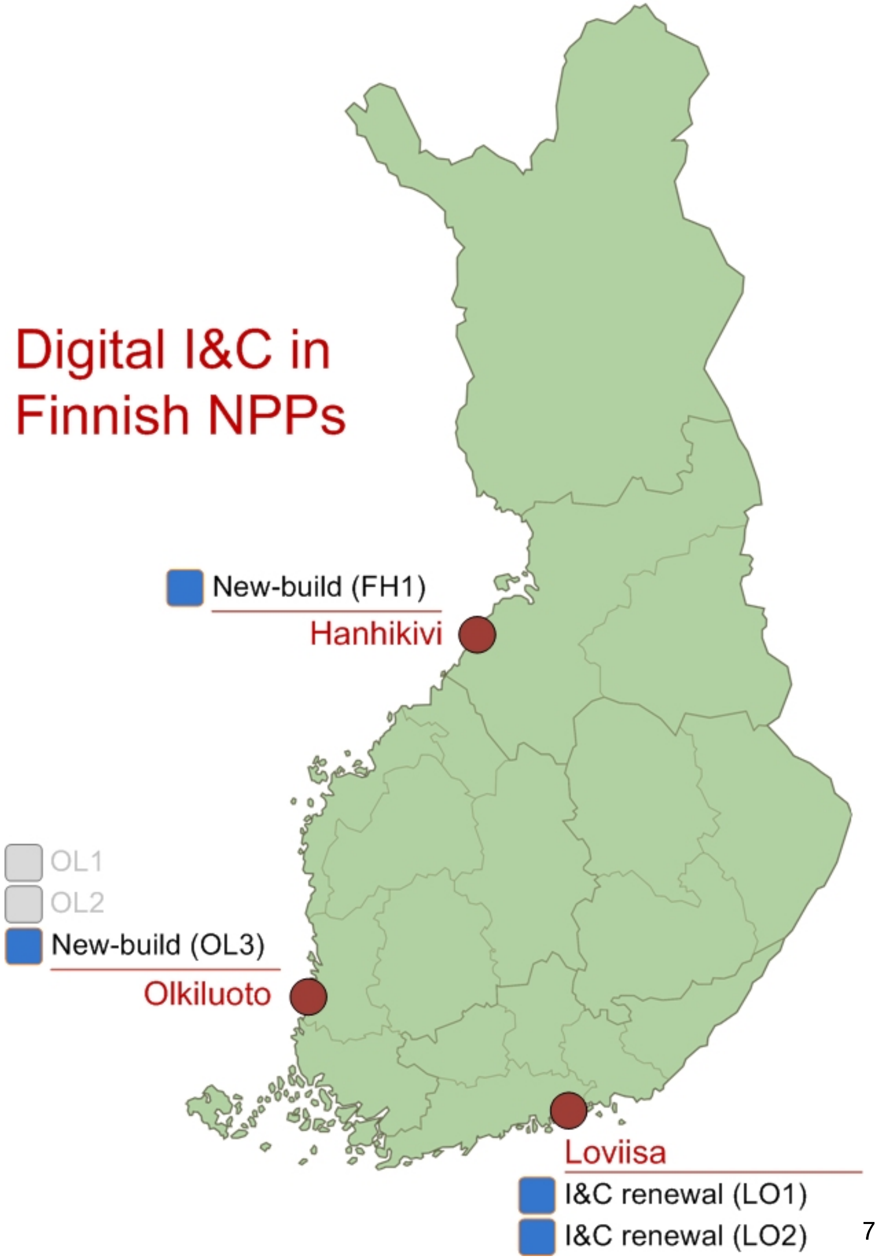
Loviisa 1 & 2 I&C modernization

- Verification of nuclear automation 2009
- Verification of nuclear I&C in the LARA project 2012-2014
- Verification of nuclear I&C in the ELSA project 2016

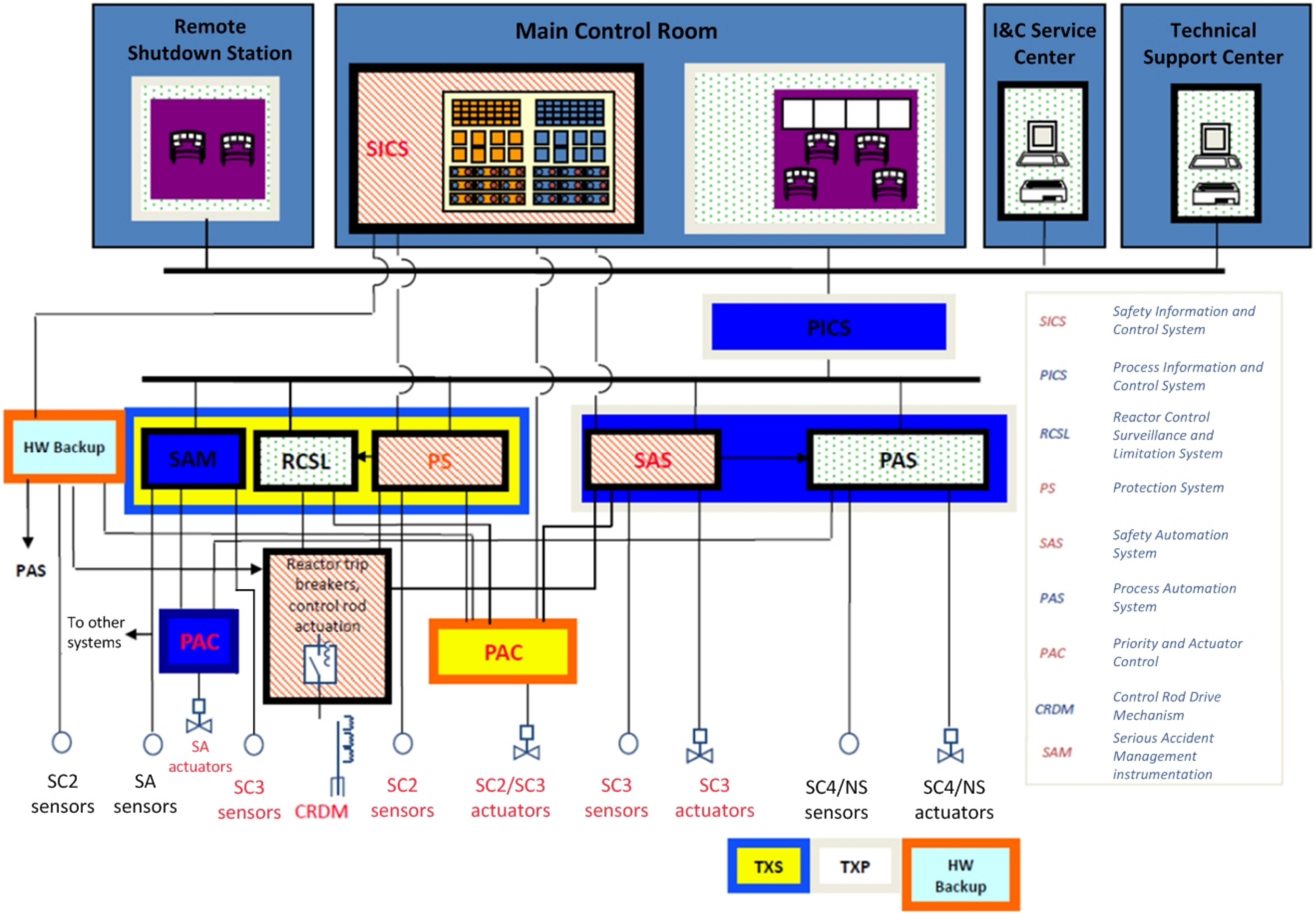
FENNOVOIMA

Hanhikivi 1 (decision-in-principle)

- Model checking of functional, architecture-level I&C 2016



OL3 I&C Architecture



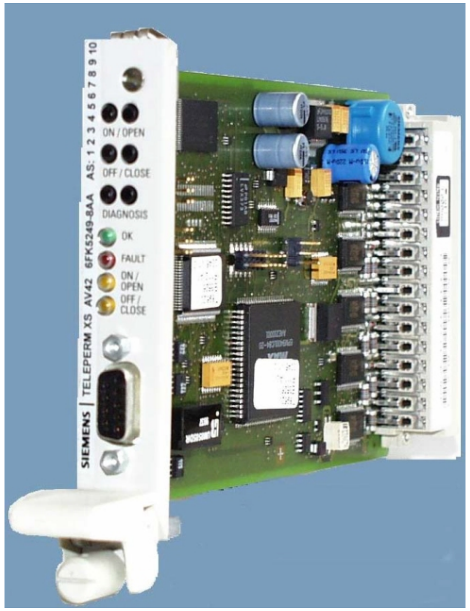
SICS	*Safety Information and Control System*
PICS	*Process Information and Control System*
RCSL	*Reactor Control Surveillance and Limitation System*
PS	*Protection System*
SAS	*Safety Automation System*
PAS	*Process Automation System*
PAC	*Priority and Actuator Control*
CRDM	*Control Rod Drive Mechanism*
SAM	*Serious Accident Management instrumentation*

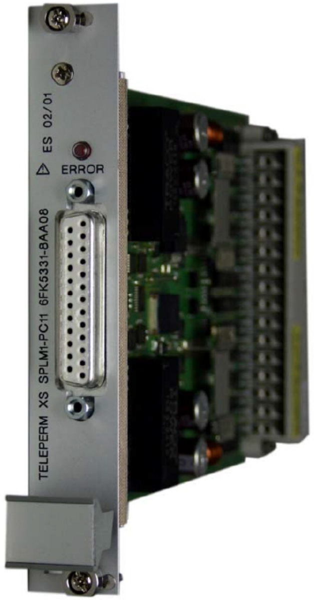
Priority and Actuator Control System (PACS) functions

- Between actuators and main I&C systems
- Controls:
  - Valves (control, isolation, solenoid)
  - Motors for various components (e.g. pumps, fans)
- Performs functions:
  - Prioritization of actuation requests
  - Drive actuation
  - Drive monitoring
  - Component protection (terminate command to valve if a travel limit and/or torque limit switch responds)

PACS components

- To introduce diversity, OL3 PACS uses two different modules:
  - AV42
  - SPLM1-PC11


SIEMENS | TELEPERM XS AV42 6FK5249-8AA AS: 1 2 3 4 5 6 7 8 9 10
ON / OPEN
OFF / CLOSE
DIAGNOSIS
OK
FAULT
ON / OPEN
OFF / CLOSE

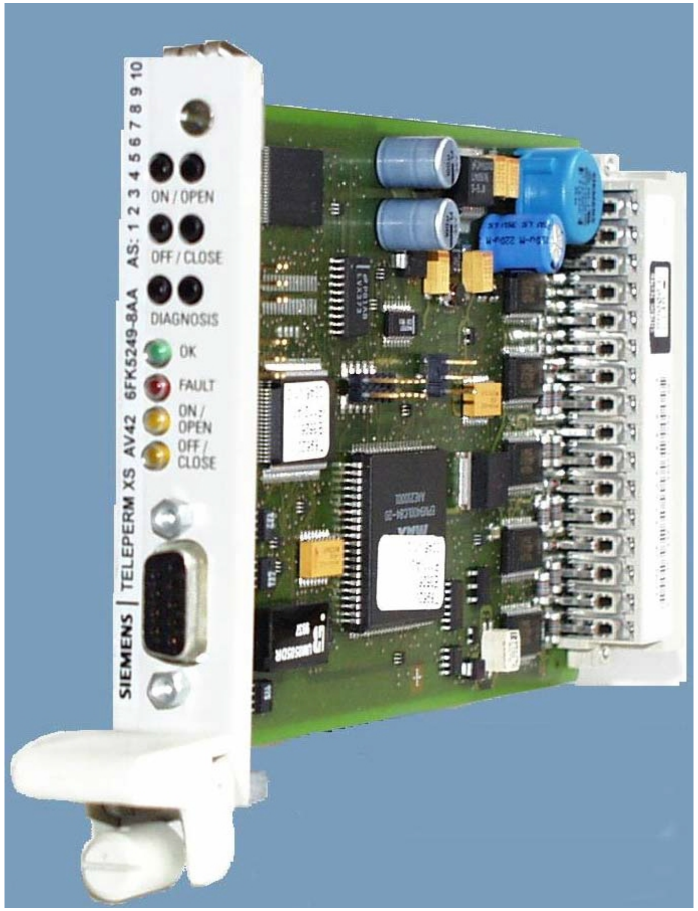


AREVA

TELEPERM XS

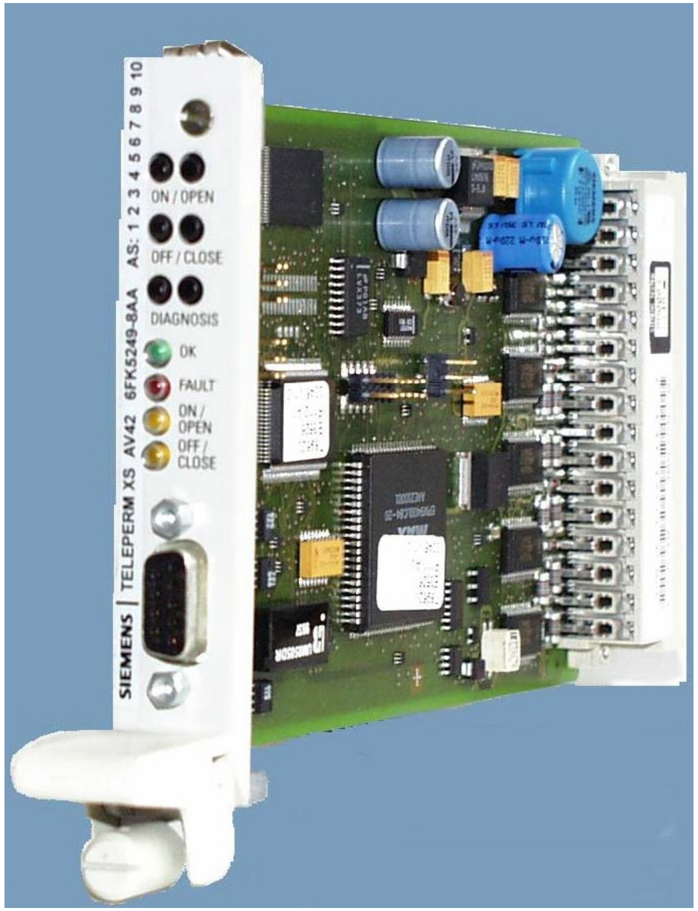
AV42

- Areva NP AV42 Priority Actuation and Control (PAC) Module
- Two major components:
  - **Programmable logic device (PLD)** consisting of interconnected logic gate arrays
  - ASIC PROFIBUS controller for non-safety-related functions
- Detailed design specification, using the ALTERA tool for PLD programming with **predefined function blocks**
- Programming tool: ALTERA MAX+plus II


SIEMENS | TELEPERM XS AV42 6FK5249-8AA AS: 1 2 3 4 5 6 7 8 9 10
ON / OPEN
OFF / CLOSE
DIAGNOSIS
OK
FAULT
ON / OPEN
OFF / CLOSE

AV42 PLD functions

- Safety-related functions implemented in the PLD:
  - Acquisition and prioritization of safety-related commands
  - Acquisition and processing of the checkback signals from the actuators
  - Command output and command termination
  - Output of signals to lamps on I&C panels
  - Test logic incl. lamp test

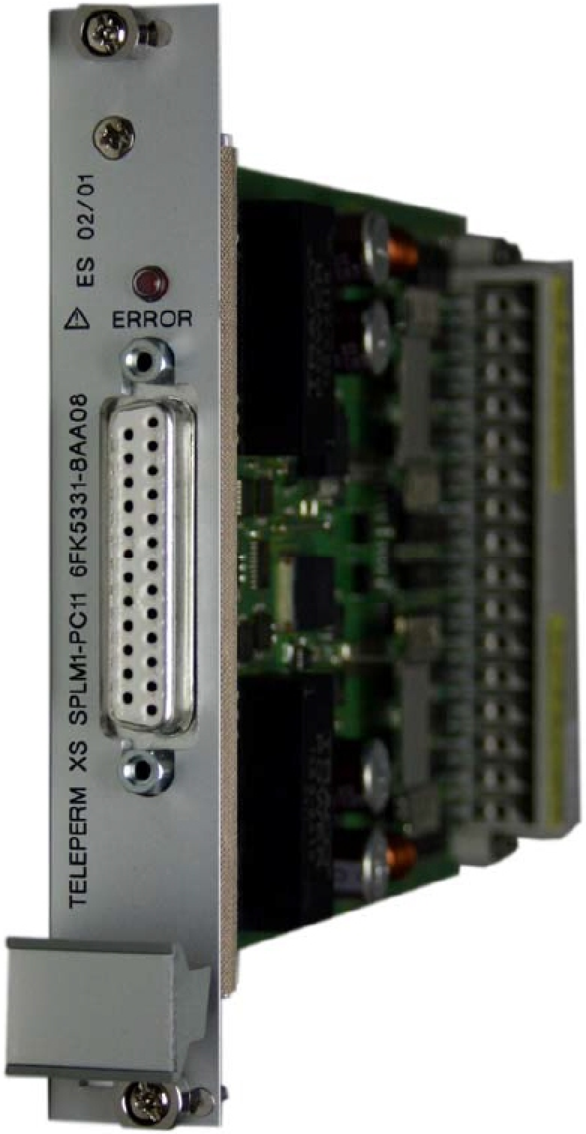

SIEMENS | TELEPERM XS AV42 6FK5249-8AA AS: 1 2 3 4 5 6 7 8 9 10
ON / OPEN
OFF / CLOSE
DIAGNOSIS
OK
FAULT
ON / OPEN
OFF / CLOSE

- Application I/O number: **60**
- Internal memories, delays

SPLM1-PC11

- Logic functions for priority and monitoring are distributed among **two PLD devices.**
- a dedicated instance of the SPLM1 programmable logic module of the TELEPERM XS (TXS) equipment platform

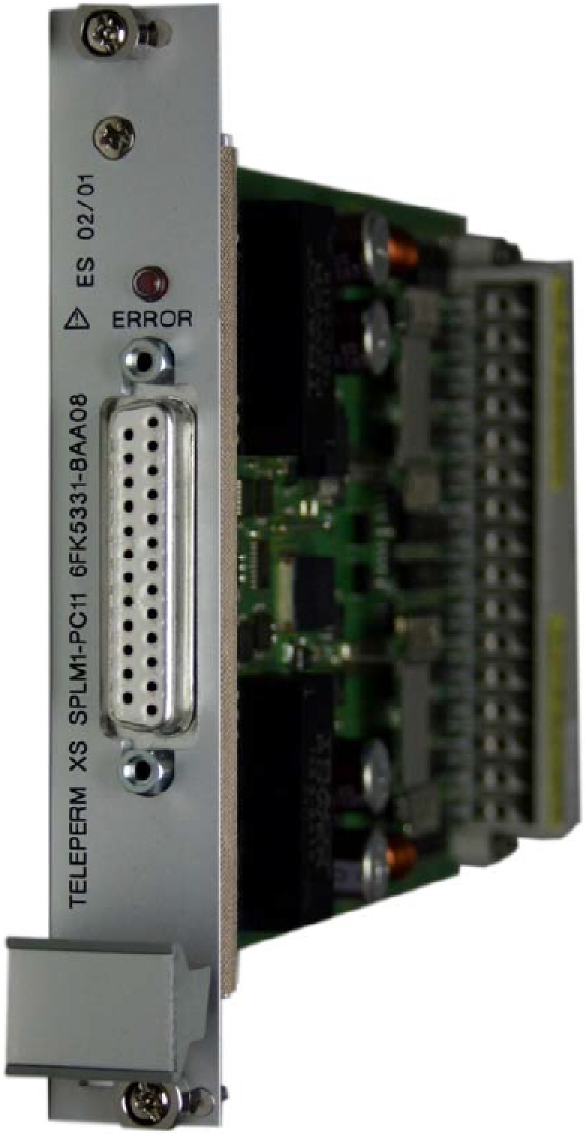
- Detailed design specification using **VHDL**



SPLM1-PC11 functions

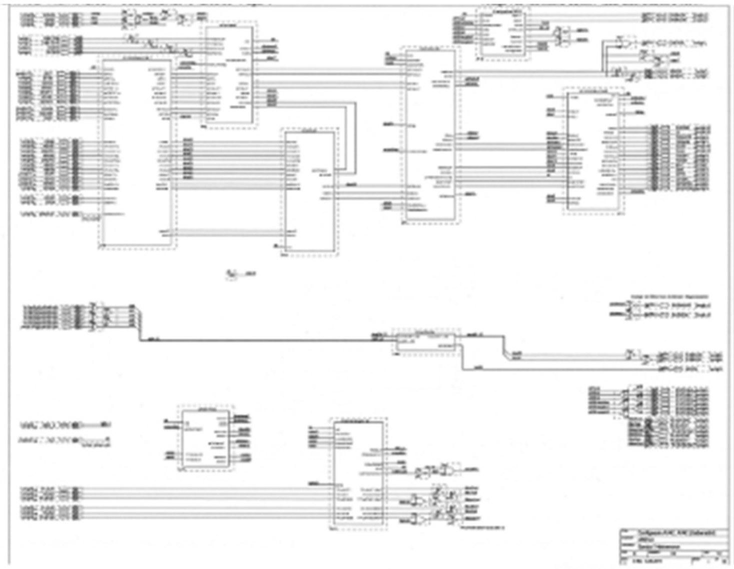
- Safety-related functions implemented in the PLDs:
    - Acquisition and prioritization of safety-related commands
    - Acquisition and processing of the checkback signals from the actuators
    - Command output and command termination
    - Output of signals to lamps on I&C panels
    - Test logic incl. lamp test

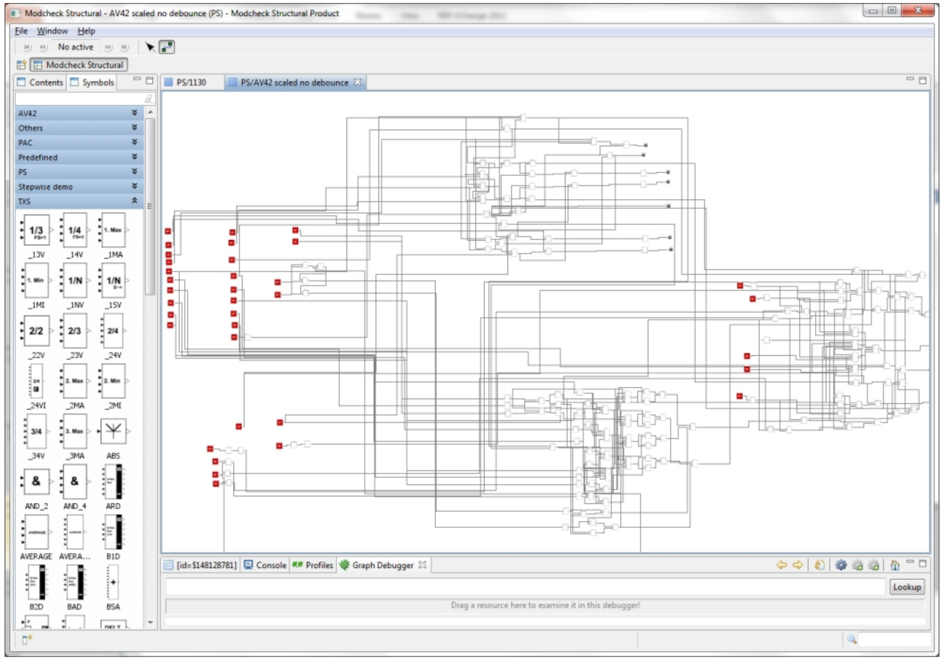
- Application I/O number: ~**40**
- VHDL code: ~2500 lines
- Internal memories, delays



Verifying AV42

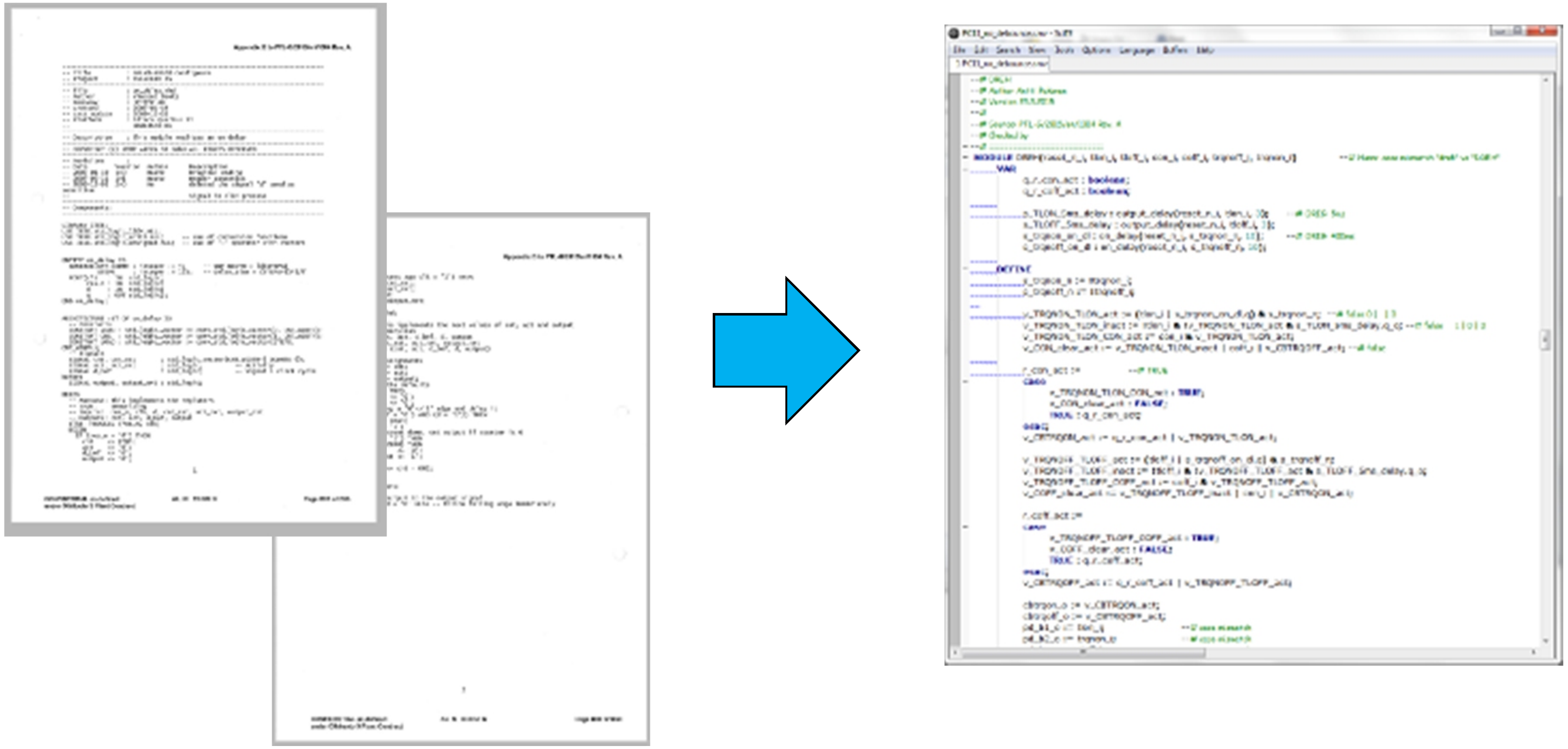
- Model built graphically, using MODCHK
- Model state space: ~ 10^{14} reachable states
- Analysis times: less than a second – some minutes (depending on property)





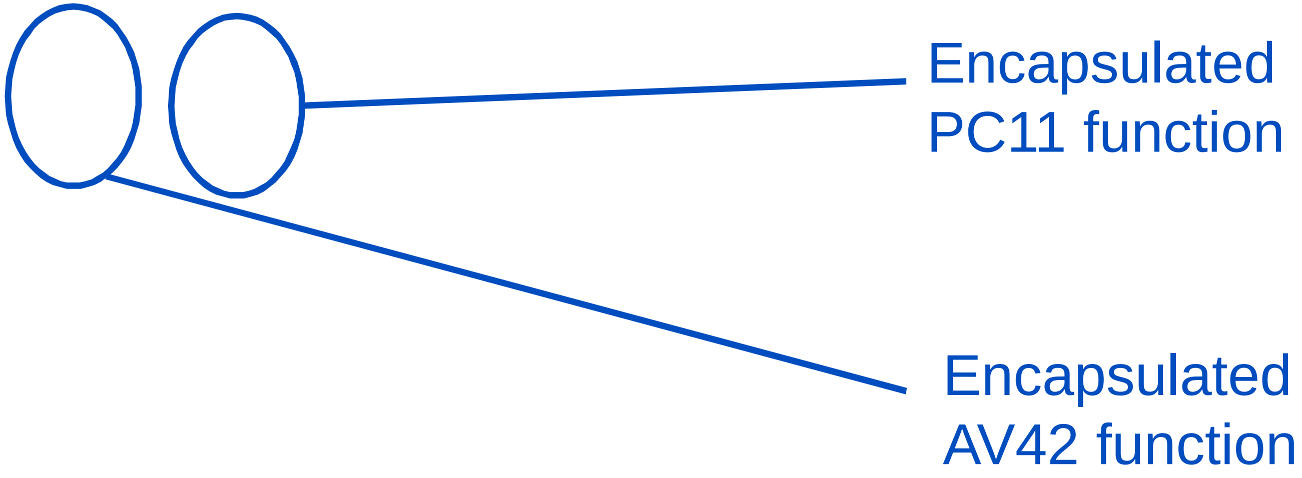
Verifying PC11

- Model built manually based on VHDL code
- Model state space: ~ 10^{20} reachable states
- Analysis times: seconds



Verifying the functionality of PS + AV42 + PC11

Confidential figure redacted.



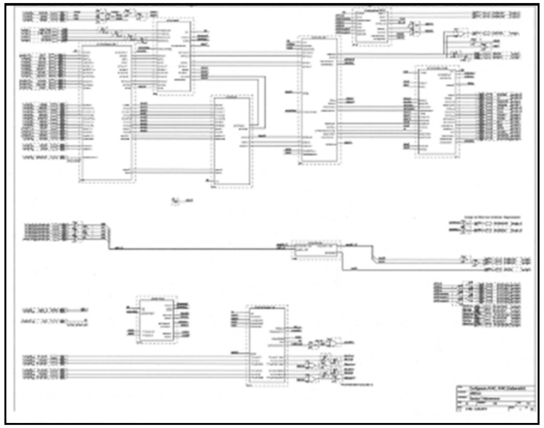
Encapsulated PC11 function

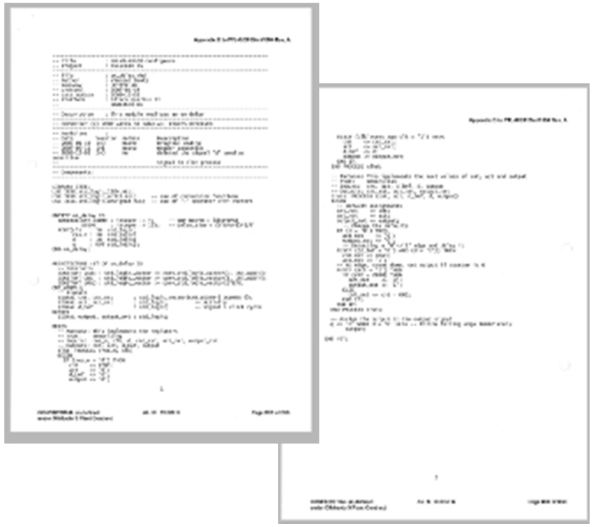
Encapsulated AV42 function

Verification results for the PLDs

- No issues relevant to safety were detected in the PLD application logics.

- Minor issues (no practical relevance) related to checkback delays – discrepancy between processing details and simplified presentation in the manuals



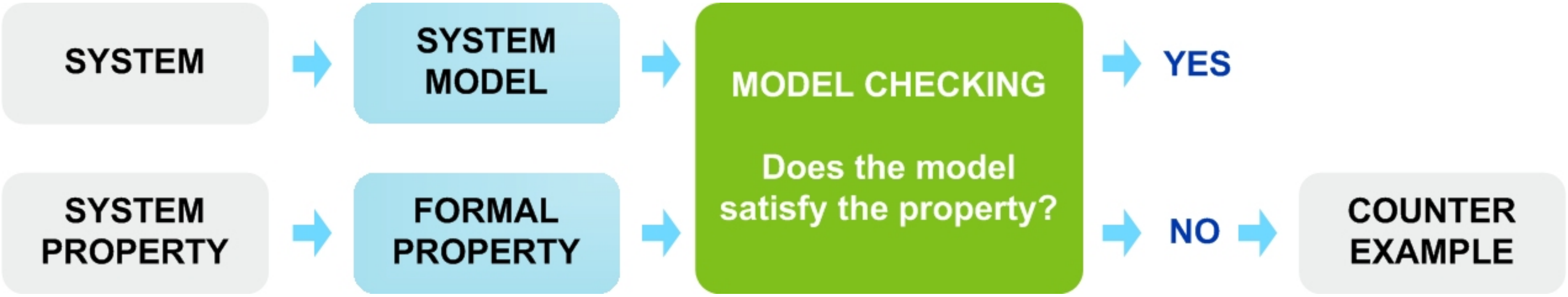


Application logic: Microprocessor vs. FPGA

Microprocessor	FPGA
The application logic is designed using function blocks, or some other suitable programming language.	The application logic is designed using function blocks, or some other suitable programming language.
It is very easy to design an application logic that is very complex.	It is very easy to design an application logic that is very complex.
Most real-world applications are complex enough to prevent 100% functional testing.	Most real-world applications are complex enough to prevent 100% functional testing.
While requiring *some* effort, model checking is a relatively cheap method given the benefits.	While requiring *some* effort, model checking is a relatively cheap method given the benefits.

Conclusions

- Model checking of FPGA designs is feasible in practice.
- Expertise is required, but work effort is calculated in days, not weeks.
- The analysis itself is *fast* and exhaustive, even when verifying systems that are consist of both microprocessor *and* FPGA application logic.
- **http://www.vttresearch.com/modelchecking**


SYSTEM
SYSTEM MODEL
SYSTEM PROPERTY
FORMAL PROPERTY
MODEL CHECKING
Does the model satisfy the property?
YES
NO
COUNTER EXAMPLE