APPLYING MODEL CHECKING TO HIGHLY-CONFIGURABLE SAFETY CRITICAL SOFTWARE: THE SPS-PPS PLC PROGRAM

B. Fernández, I. D. Lopez-Miguel, J-C. Tournier, E. Blanco, T. Ladzinski, F. Havart, CERN, Geneva, Switzerland

WEPV042

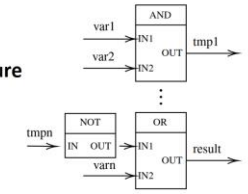
ICALEPCS 2021

THE SPS PERSONNEL PROTECTION SYSTEM (PPS)

SPS-PPS Hardware architecture
PLC-based control system
S7-1516F

- **Highly** distributed and **configurable** system
- **Site PLCs**: common PLC program and different configurations per access zone

SPS-PPS Site PLC program architecture
Set of *Functions (FC)* and *Function Blocks (FBs)*



- **TIA Portal** programming environment
- Function Block Diagram (**FBD**) **programming language**
- Complex and large PLC logic
- Complex specifications

THE SITE SPS-PPS PLC PROGRAM AND SPECIFICATIONS

Complex requirement specifications based on
if-else* conditional statement that contains *Boolean formulas

$$
\begin{aligned}
&\textbf{if } (\\
&\bigwedge_{i=1}^{N_i} (I\_EISa\_Pos\_Stat[i] \vee I\_EISa\_Bypass[i]) \\
&\forall i \in [1, N_i] (I\_EISa\_Pos[i]=1 \wedge SC\_S[j][i]=1 \\
&\wedge \\
&\bigwedge_{i=1}^{N_i} (I\_EISa\_PU\_Stat[i] \vee I\_EISa\_Bypass[i]) \\
&\forall i \in [1, N_i] (I\_EISa\_PU[i]=1 \wedge SC\_S[j][i]=1 \\
&\wedge \\
&S0\_AP\_Pos[j] \wedge S0\_AP\_PU[j] \wedge S0\_AP\_Key\_Distrib[j] \\
&) \textbf{ then} \\
&\quad N\_EISa\_Safe[j] \leftarrow 1 \\
&\textbf{else} \\
&\quad N\_EISa\_Safe[j] \leftarrow 0 \\
&\textbf{end if}
\end{aligned}
$$

Example of SIF-X1 behavior

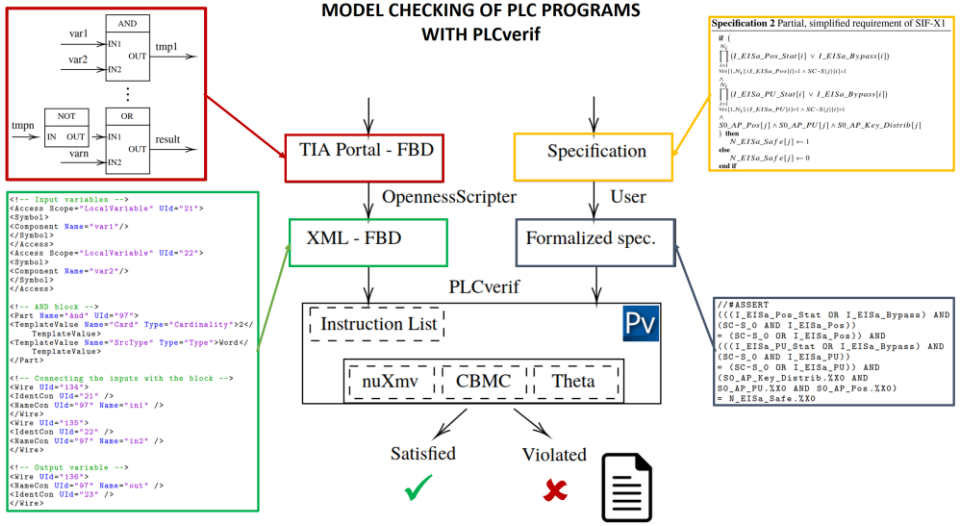
Source	Variable	Value
Input	$EISa\_Pos\_Stat$	0000 0000 0000 1001
Input	$EISa\_PU\_Stat$	0000 0000 0000 0001
...	...	...
Input (AP PLC)	$S0\_AP\_Pos$	0000 0000 0000 1001
...	...	...
Configuration	$EISa\_Pos$	0000 0000 0000 1001
Configuration	$EISa\_PU$	0000 0000 0000 0001
...	...	...
Configuration	SC_S_0	0000 0000 0000 0001
Configuration	SC_S_1	0000 0000 0000 1000
...	...	...
Output	N_EISa_Safe	0000 0000 0000 1001

Lots of configuration variables (SIF-X1 **example**)

Configuration variables	94 WORD (16-bit) and 4 BOOL
Input variables	21 WORD (16-bit) and 2 BOOL
Potential State Space (nb. of combinations to be checked)	$5.0 \cdot 10^{555}$

Impossible to validate the PLC program **only** with **testing**
Model checking is recommended by the Functional Safety standards

MODEL CHECKING OF PLC PROGRAMS WITH PLCverif



Model Checking (MC) is recommended by the **Functional Safety standards** (IEC 61508 and IEC 61511)

Conclusions of using MC in this project	Future work
detect bugs in the **PLC program** before commissioning	**integration** of model checking in PLC programming environments (e.g. TIA portal)
identify deficiencies in the **specification**	improvement of **verification performance** with better algorithms and abstraction techniques
help experts to better **understand** the **behaviour** of the program for all possible configurations (**corner cases**)	

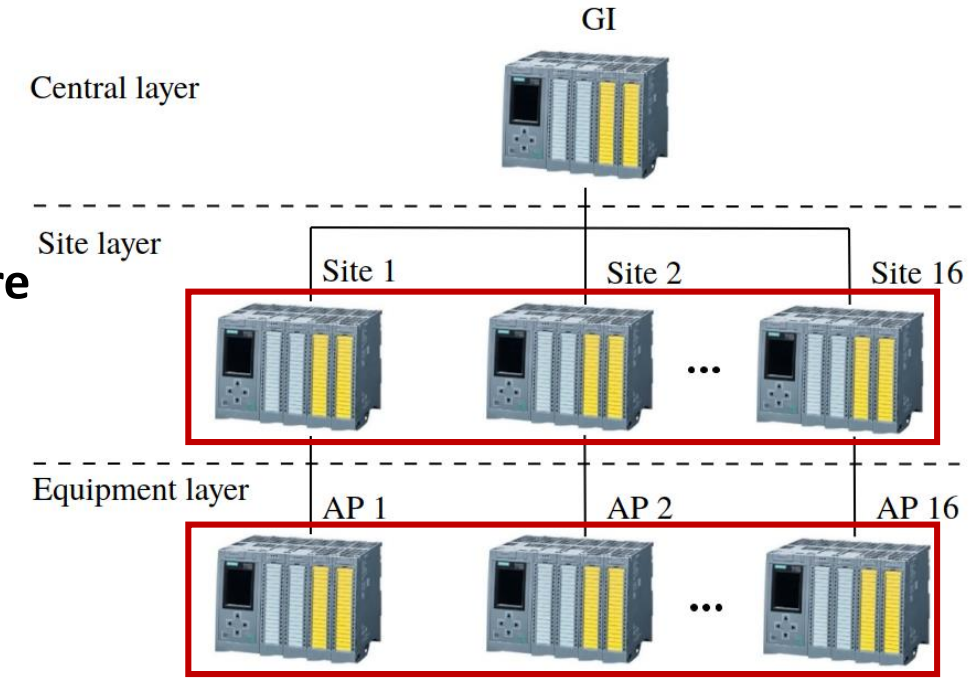
PLCverif allows to apply model checking to PLC program and **hides the complexity associated to formal methods**

THE SPS PERSONNEL PROTECTION SYSTEM (PPS)

SPS-PPS Hardware architecture

PLC-based control system

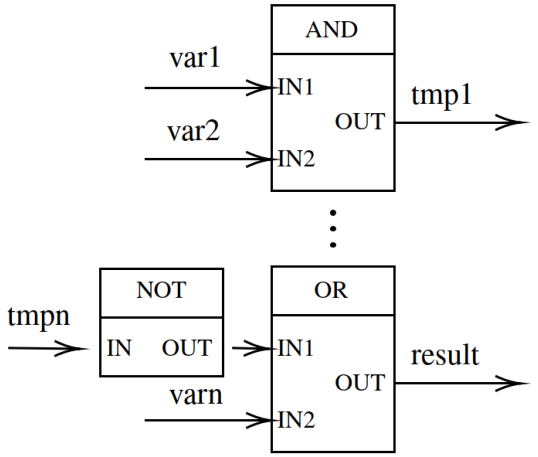
S7-1516F



- **Highly** distributed and **configurable** system
- **Site PLCs**: common PLC program and different configurations per access zone

SPS-PPS Site PLC program architecture

Set of *Functions (FC)* and *Function Blocks (FBs)*



TIA V16

- **TIA Portal** programming environment
- Function Block Diagram (**FBD**) **programming language**
- Complex and large PLC logic
- Complex specifications

THE SITE SPS-PPS PLC PROGRAM AND SPECIFICATIONS

Complex requirement specifications based on

if-else* conditional statement that contains *Boolean formulas

$$
\begin{aligned}
&\textbf{if } \Big(\\
&\prod_{i=1}^{N_i} (I\_EISa\_Pos\_Stat[i] \vee I\_EISa\_Bypass[i]) \\
&\forall i \in [1, N_i]: (I\_EISa\_Pos[i]{=}1 \wedge SC{-}S\{j\}[i]{=}1 \\
&\wedge \\
&\prod_{i=1}^{N_i} (I\_EISa\_PU\_Stat[i] \vee I\_EISa\_Bypass[i]) \\
&\forall i \in [1, N_i]: (I\_EISa\_PU[i]{=}1 \wedge SC{-}S\{j\}[i]{=}1 \\
&\wedge \\
&S0\_AP\_Pos[j] \wedge S0\_AP\_PU[j] \wedge S0\_AP\_Key\_Distrib[j] \\
&\Big) \textbf{ then} \\
&\quad N\_EISa\_Safe[j] \leftarrow 1 \\
&\textbf{else} \\
&\quad N\_EISa\_Safe[j] \leftarrow 0 \\
&\textbf{end if}
\end{aligned}
$$

Example of SIF-X1 behavior

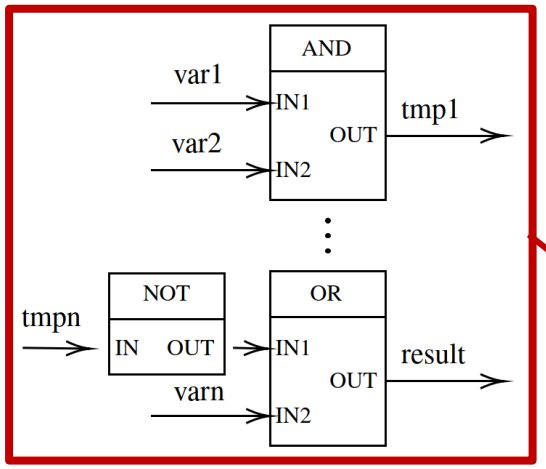
Source	Variable	Value
Input	$EISa\_Pos\_Stat$	0000 0000 0000 1001
Input	$EISa\_PU\_Stat$	0000 0000 0000 0001
...	...	...
Input (AP PLC)	$S0\_AP\_Pos$	0000 0000 0000 1001
...	...	...
Configuration	$EISa\_Pos$	0000 0000 0000 1001
Configuration	$EISa\_PU$	0000 0000 0000 0001
...	...	...
Configuration	$SC - S\_0$	0000 0000 0000 0001
Configuration	$SC - S\_1$	0000 0000 0000 1000
...	...	...
Output	N_EISa_Safe	0000 0000 0000 1001

Lots of configuration variables (SIF-X1 **example**)

Configuration variables	94 WORD (16-bit) and 4 BOOL
Input variables	21 WORD (16-bit) and 2 BOOL
Potential State Space (nb. of combinations to be checked)	$5.0 \cdot 10^{555}$

Impossible to validate the PLC program **only** with **testing**
Model checking is recommended by the Functional Safety standards

MODEL CHECKING OF PLC PROGRAMS WITH PLCverif




```
<!-- Input variables -->
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="var1"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="var2"/>
</Symbol>
</Access>

<!-- AND block -->
<Part Name="And" UId="97">
<TemplateValue Name="Card" Type="Cardinality">2</
    TemplateValue>
<TemplateValue Name="SrcType" Type="Type">Word</
    TemplateValue>
</Part>

<!-- Connecting the inputs with the block -->
<Wire UId="134">
<IdentCon UId="21" />
<NameCon UId="97" Name="in1" />
</Wire>
<Wire UId="135">
<IdentCon UId="22" />
<NameCon UId="97" Name="in2" />
</Wire>

<!-- Output variable -->
<Wire UId="136">
<NameCon UId="97" Name="out" />
<IdentCon UId="23" />
</Wire>
```

TIA Portal - FBD

OpennessScripter

XML - FBD

Specification

User

Formalized spec.

PLCverif

Instruction List

nuXmv | CBMC | Theta

Satisfied

Violated

Specification 2 Partial, simplified requirement of SIF-X1

if (

$$\prod_{i=1}^{N_I} (I\_EISa\_Pos\_Stat[i] \vee I\_EISa\_Bypass[i])$$

$\forall i \in [1,N_I]: (I\_EISa\_Pos[i]=1 \wedge SC\text{-}S\{j\}[i]=1$

$\wedge$

$$\prod_{i=1}^{N_I} (I\_EISa\_PU\_Stat[i] \vee I\_EISa\_Bypass[i])$$

$\forall i \in [1,N_I]: (I\_EISa\_PU[i]=1 \wedge SC\text{-}S\{j\}[i]=1$

$\wedge$

$S0\_AP\_Pos[j] \wedge S0\_AP\_PU[j] \wedge S0\_AP\_Key\_Distrib[j]$

) **then**

$N\_EISa\_Safe[j] \leftarrow 1$

else

$N\_EISa\_Safe[j] \leftarrow 0$

end if

```
//#ASSERT
(((I_EISa_Pos_Stat OR I_EISa_Bypass) AND
(SC-S_0 AND I_EISa_Pos))
= (SC-S_0 OR I_EISa_Pos)) AND
(((I_EISa_PU_Stat OR I_EISa_Bypass) AND
(SC-S_0 AND I_EISa_PU))
= (SC-S_0 OR I_EISa_PU)) AND
(S0_AP_Key_Distrib.%X0 AND
S0_AP_PU.%X0 AND S0_AP_Pos.%X0)
= N_EISa_Safe.%X0
```

Model Checking (MC) is recommended by the **Functional Safety standards**
(IEC 61508 and IEC 61511)

Conclusions of using MC in this project	Future work
detect bugs in the **PLC program** before commissioning	**integration** of model checking in PLC programming environments (e.g. TIA portal)
identify deficiencies in the **specification**	improvement of **verification performance** with better algorithms and abstraction techniques
help experts to better **understand** the **behaviour** of the program for all possible configurations (**corner cases**)	

PLCverif allows to apply model checking to PLC program and
hides the complexity associated to formal methods