Formal verification of embedded logic controller specification with computer deduction in temporal logic

Abstract. The article presents a novel approach to formal verification of logic controller specification. Model checking technique is used to verify some behavioral properties. The approach proposes to use a rule-based logical model presented at RTL-level. Proposed logical model is suitable both for formal verification (model checking in the NuSMV tool) and for logical synthesis (using hardware description language VHDL). As the result, logic controller program (its implementation) will be valid according to its primary specification.

1. Introduction

Embedded logic controllers specification is the first step in development process. Possible errors in this phase [1] may influence oncoming phases or even the whole venture. Usually, it generates enormous costs to remove errors which were detected too late. Dependable embedded logic controllers have additional requirements which requires beside high quality also reliability, availability, safety and secureness. Even a tiny error in design phase may change the total system behavior and may have tragic effects.

One of commonly used formal techniques by logic controllers specification are Petri Nets (PN) [2,3], and especially Control Interpreted Petri Nets (CIPN) [3,4]. They are well suited for modeling of hardware behavior including i.e. concurrency or resources sharing. There are many approaches to analyze CIPNs and check their boundness or liveness. There are however few approaches which allow to formally verify Petri Nets against some defined requirements, and none of them addresses directly CIPNs focused on RTL-level and their behavioral specification. Model checking technique with temporal logic offers the possibility to check behavior of designed logic controller.

Model checking [5] is one of formal verification methods among others like e.g. theorem proving [6,7] and is currently used in the industry in hardware [8] and software production. System model is compared with defined properties and an answer whether they are satisfied or not is given. In case of detected errors, appropriate counterexamples are generated which allow to localize error source.

The article is structured as follows. Section 2 describes related work in the area. Section 3 presents novel approach to formal verification of embedded logic controller specification with computer deduction in temporal logic. Embedded logic controller specification is formally written as an Control Interpreted Petri Net. The approach proposes to use a rule-based logical model presented at RTL-level (subsection 3.1) suitable both for formal verification (subsection 3.2) and for logical synthesis (subsection 3.3). Section 4 concludes the paper.

2. Related work

There have already been some approaches to verify Petri Nets. However, none of them addressed Control Interpreted Petri Nets focused on RTL-level and checked against behavioral properties. There have also been some approaches to verify UML diagrams, like in [9,10]. Programs for PLC controllers [11] have also been verified, prepared as the RT textual language [12], the Grafnet language (sequential part) [13] or the SFC graphical and hierarchical language [14].

In [15] Timed Petri Nets with clocks assigned to transitions or places are taken into account. Timed Petri Nets are also considered in [16] (nets with additional priorities). In [17] Petri Nets with time stamps for embedded systems are considered. Tokens hold there a value and a time stamp, what makes the nets different from classical Petri Nets. Furthermore, each transition is interpreted as a separate process, which changes marking of places.

In [18] Synchronous and Interpreted Petri Nets are considered. Authors verify them using PROMELA language and the SPIN model checker. In the first approach they do not allow for two or more guards of transitions (input signals of logic controller) to be simultaneously true, what makes difficult to analyze system behavior. In the second approach they also focus more on structural properties than on the behavioral. Priorities for transitions are introduced artificially, what causes deformation of system functionality. It is also hardly acceptable, that only one input signal can be active at one moment.

Petri Nets verification has also been proposed by G. Frey. In [19] a complete tool for logic controllers development with usage of Petri Nets is presented. Authors focused on Signal Interpreted Petri Nets (SIPN), which are similar to Control Interpreted Petri Nets (CIPN). The end result of logic controller development is here a program for PLC. Proposed approach does not take into account reconfigurable logic controllers based on FPGA circuits.

3. Novel approach to formal verification of embedded logic controller specification

The novel approach to model checking of Control Interpreted Petri Nets focuses on rule-based logical model generation basing on a CIPN. Logical model with temporal logic formulas is presented at RTL-level in such a way that it is easy to synthesize as reconfigurable logic controller or PLC as well as to formally verify for behavioral properties (see Fig. 1). Logical model is therefore transformed into model description in the NuSMV model checker [20] input language with behavioral assertions. Model checker tool verifies the model and checks whether some defined behavior properties are fulfilled. On the other hand, logical model is transformed into hardware description language (i.e. VHDL) with structural assertions. Logical model is
hence used for synthesis purposes as well as for model checking. It is a format in which the behavior of logic controller is specified. It contains logical formulas describing changes in the designed system.

![Control Interpreted Petri Net](image)

**Model checking**

**Logical model**

**Logical synthesis**

### 3.1 Logical model

Logical model includes variables definition with their initial values. The following elements of Control Interpreted Petri Net are interpreted as model variables:

#### a) places (P)

Each place of Control Interpreted Petri Net is assigned a variable of Boolean type. It takes the TRUE value if the place is marked. It is possible that multiple places have the TRUE value assigned, as there may be several concurrent processes defined.

#### b) input signals (X)

Each input signal of logic controller is assigned a variable of Boolean type. It takes the TRUE value if the signal is active and the FALSE value if it is inactive. It is possible that multiple input signals have the TRUE value assigned, as there may be several input signals active at the same time.

#### c) output signals (Y)

Each output signal of logic controller is assigned a variable of Boolean type. It takes the TRUE value if the signal is active and the FALSE value if it is inactive. It is possible that multiple output signals have the TRUE value assigned, as there may be several output signals active at the same time.

#### d) Defined variables take predefined initial values:

- Each place \( p \in P \) is a variable of Boolean type
- For each variable \( p \in \text{places} \) a separate variable \( p : \text{boolean} \) is defined
- Each input signal \( x \in X \) is a variable of Boolean type
- For each variable \( x \in \text{inputs} \) a separate variable \( x : \text{boolean} \) is defined
- Each output signal \( y \in Y \) is a variable of Boolean type
- For each variable \( y \in \text{outputs} \) a separate variable \( y : \text{boolean} \) is defined

#### e) Each place changes its marking according to the predefined rules:

- \( \text{init(var)} := \text{TRUE} \) or \( \text{init(var)} := \text{FALSE} \)
- Each place changes its marking according to the defined rules; conditions of changes between places (token flow) occur in pairs – in the previous place and in the next place.

#### f) Each output signal changes its value according to the predefined rules

### 3.2 Formal verification

Logical model derived from Control Interpreted Petri Net is transformed into format of the NuSMV model checker according to the following rules:

- Each place \( p \in P \) is a variable of Boolean type
- For each variable \( p \in \text{places} \) a separate variable \( p : \text{boolean} \) is defined
- Each input signal \( x \in X \) is a variable of Boolean type
- For each variable \( x \in \text{inputs} \) a separate variable \( x : \text{boolean} \) is defined
- Each output signal \( y \in Y \) is a variable of Boolean type
- For each variable \( y \in \text{outputs} \) a separate variable \( y : \text{boolean} \) is defined

The following code presents some snapshots from a sample logical model based on a Control Interpreted Petri Net (part of it in Fig. 2):
g) Each input signal changes randomly, but can take the expected values connected with Petri net places or change adequately to the situation. Changing of variables values are considered as separate segments in the NuSMV:

\[
\text{next}(\text{variable}) := \text{case}
\]
\[
\text{deactivation\_condition} : \text{FALSE};
\]
\[
\text{activation\_condition} : \text{TRUE};
\]
\[
\text{TRUE} : \text{variable};
\]
\[
esac;
\]

Properties to be checked are defined using either LTL or CTL logic [21-24]. Properties [25] describe safety requirements (something bad will never happen), as well as liveness requirements (something good will eventually happen). Safety and liveness requirements are the most frequently specified requirements to be verified. The requirements list should include as much desired properties as possible, as only they will be checked.

Structural properties can be checked on the Petri net level and does not require using model checking technique. Properties like liveness or deadlock-freedom can be verified by some tools dedicated to Petri nets, like i.e. WoPeD [26] or PIPE2 [27].

Properties are defined according to the schema:

\[
\text{LTLSPEC} \text{ LTL\_expression} \text{ or CTLSPEC} \text{ CTL\_expression}
\]

A sample liveness property is \(\text{LTLSPEC} \ G \ (y_4 \land x_4 \rightarrow F \ (y_5 \land y_6))\) what means, that always when the \(y_4\) output signal is active and the \(x_4\) input signal becomes active, finally both output signals \(y_5\) and \(y_6\) will be active.

A sample safety property is \(\text{LTLSPEC} \ !\ (y_5 \land y_8)\) what means, that it should never be the case, that both output signals \(y_5\) and \(y_8\) are active at the same time.

The following code presents some snapshots from model description in the NuSMV tool:

```vhdl
VAR
  p1 : boolean;
p2 : boolean;
...
x0 : boolean;
x1 : boolean;
...
y0 : boolean;
y1 : boolean;
...
ASSIGN
  init(p1) := TRUE;
  init(p2) := FALSE;
  ...
  init(x0) := FALSE;
  init(x1) := FALSE;
  ...
  init(y0) := TRUE;
  init(y1) := FALSE;
  ...
next(p1) := case
  p1 & x0 : FALSE;
p6 & p9 & !x6 : TRUE;
TRUE : p1;
esac;
...
next(y1) := case
  p2 : TRUE;
TRUE : FALSE;
esac;
...
next(x0) := case
  p1 : (FALSE, TRUE);
TRUE : FALSE;
```
use another forms of system specification as basis for formal verification and synthesis, i.e. UML 2.x Activity Diagrams which can be transformed into CIPNs [29].

Presented approach was tested on several examples of industrial logic controllers specifications by means of Control Interpreted Petri Nets. As a support for testing, a tool has been developed, which allows automatic transformation from logical model into model description in the NuSMV format. Transformation into VHDL code is still under development.

Results of the work include the assurance that verified behavioral specification in temporal logic will be an abstract program of matrix reconfigurable logic controller. So, logic controller program (its implementation) will be valid according to its primary specifications.

ACKNOWLEDGMENTS

The author is a scholar within Sub-measure 8.2.2 Regional Innovation Strategies, Measure 8.2 Transfer of knowledge, Priority VIII Regional human resources for the economy Human Capital Operational Programme co-financed by European Social Fund and state budget

BIBLIOGRAPHY


Authors: mgr. inż. Iwona Grobelna, University of Zielona Gora, ul. Podgorna 50, 65-246 Zielona Gora, Poland E-mail: i.grobelna@iie.uz.zgora.pl