Efficient Design-Flow for Counting Heads

Sebastian Kinder Rolf Drechsler
Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
{kinder,drechsle}@informatik.uni-bremen.de

I. Introduction

Nowadays railway systems are designed and tested in a conventional way, i.e. the systems are simulated with a manually created test bench. This has the advantage that the designers have a considerable expertise with this kind of work, but there is still a lot of potential for human failure. Furthermore, testing is very cost-intensive and can never reach complete coverage. Hence, an integrated design flow for railway systems is needed, which allows for efficient modeling, validation, simulation-based verification as well as formal verification. A well accepted approach for this is the system level description language SystemC, which is known from the hardware design domain. SystemC supports the designer with a suitable methodology for efficient modeling and validation. With an integrated simulation kernel it facilitates a fast and efficient simulation-based validation and verification.

In this paper we present a design flow for a railway specific application based on SystemC. We show the modeling of *Counting Heads* (CHs) [7] for railways, which are used to determine whether a specified *Track Vacancy Detection Section* (TVDS) is clear or occupied. Especially for electronic railway interlocking systems as constructed by SIEMENS, which determine automatically, whether a TVDS is clear or occupied, the correct function of CHs is crucial: If they fail to work properly a TVDS would either be falsely indicated as occupied – resulting in a deterioration of availability and reliability – or falsely indicated as clear – possibly introducing a safety hazard. The proof of correctness helps to avoid such situations. Therefore, we present first steps towards a complete formal verification of CHs using *Bounded Model Checking* (BMC).

II. Preliminaries

A. Specification Languages

To specify the model the system description language SystemC [6] is used. A SystemC-model, which is modeled on the register transfer level (RTL), can be verified by using Property Checking. These properties can be specified using temporal expressions. A widely known industrial standard is PSL [1] from Accellera.

PSL-properties as they are used here consist of two parts: a list of assumptions and a list of commitments. They state that if all assumptions hold, all commitments have to hold as well. Otherwise the property would fail. Properties argue over a finite time interval, which is called *observation window*. In PSL there are multiple operators to allow for expressing time constraints and logical or arithmetic constraints.

B. Bounded Model Checking

In *model checking* properties for a given system are formulated in a dedicated "verification language". It is then formally proven whether these properties hold under all input and state assignments for the given assumptions. While "classical" CTL-based model checking [3] can only be applied to medium sized designs, approaches based on *Bounded Model Checking* (BMC) as discussed in [2] give very good results when used for complete blocks. In BMC the properties are only considered over a finite time interval. BMC has originally been proposed for circuit verification and in this context considering a finite number of steps is reasonable. The underlying techniques are outlined below.

The general structure of the resulting BMC instance for a property p over the finite interval $[0, c]$ is given by

$$\bigwedge_{i=0}^{c-1} T_\delta(s_i, s_{i+1}) \ \wedge \neg \, p,$$

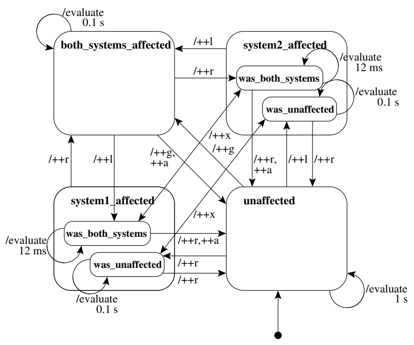


Fig. 1. FSM of the Counting Head [7]

where $T_\delta(s_i, s_{i+1})$ denotes the transition relation between cycles i and $i+1$. If this formula is satisfiable a counter example has been found, disproving the validity of p. This problem can be formulated as a SAT problem by unrolling the circuit for c time frames and generating logic for the property. Since there is no restriction to reachable states during the proof of the corresponding SAT instance a counter-example may start from an unreachable state. Usually, if such a case occurs these states are excluded by additional assumptions. Finally, if the SAT instance is unsatisfiable the property holds.

III. Modeling the Counting Heads

Counting Heads (CHs) [7] are needed to determine whether a railway track section is vacant or occupied. This is essential for electronic railway interlocking systems in order to position points into the correct direction for the next train. CHs are used in a lot of interlocking systems all over the world. A CH failing to work properly could result in a collision of railways and endanger the life of passengers.

A CH has three Boolean inputs. One of these inputs is a reset, which sets all internal variables and all outputs to 0. The other two Boolean inputs are retrieved from a *Double Wheel Detector* (DWD), which is mounted on one of the rails and detects the passing and the direction of this passing of wheels of a vehicle. The CH itself is implemented as a *finite state machine* (FSM) (see Figure 1), consisting of four states. Two of these states consist of 2 substates. This FSM is traversed corresponding to the impacts on the sensor of the DWD. Either the first sensor, both sensors, the second sensor or none of the sensors are affected. For each state transition several integer counters are manipulated. The system can be idle in each of the states/substates for a specific time, before the internal counters are evaluated. When the counters are evaluated, the output values are written. The timeouts vary from a few milliseconds up to one second. Furthermore, the system detects failures at the time of evaluation and indicates them by the corresponding outputs. Depending on the type, CHs can have different outputs, but there are four outputs for all types:

1) The number of axles is the signed sum of all axles which crossed the double wheel detector. The sign indicates the direction of the axles.
2) The number of errors is incremented every time an error occurred in the CH, e.g. an undefined counting impulse.
3) The time the CH was not affected, states for how long no axle crossed the double wheel detector.
4) The counter control token is set, if the CH raises suspicions of malfunctions.

Additionally, there can be several failure tokens, which indicate if a failure occurred, depending on the type of the CH.

IV. Simulation-based Validation

In this section we show the simulation-based validation of the SystemC model of the CH. Since SystemC is a C++

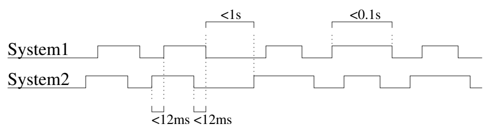
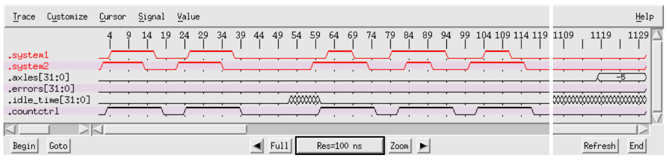


Fig. 2. Stimuli Waveform



(a) Part 1 (b) Part 2

Fig. 3. Waveform of a Simulation with a Tolerated Number of Interferences

class library, the modelled system can be compiled to an executable specification. This specification can be simulated with the integrated SystemC simulation kernel. To simulate a design it is necessary to stimulate its inputs. This is done by a special module, the stimuli generator, which is connected to all inputs of the SystemC model. The generator has exactly the same number of outputs as the design's number of inputs. These outputs have to be of the same type as the inputs of the system. For the CH-model introduced in Section III a stimuli generator with three Boolean outputs is needed to stimulate each one of the inputs `system1`, `system2` and `reset`.

In Figure 2 the stimuli for the inputs is given. The delay between two rising and falling edges respectively on the two sensor systems of the DWD is not allowed to be greater than 12 ms. No high edge may last 0.1 s or longer. Both sensor systems may not be unaffected for 1 s or more at the same time. If any of these happens, the train is considered standing and thus, the evaluation phase of the CH begins.

The stimuli from Figure 2 are produced by a stimuli generator. This waveform can be found again in the resulting waveform of the test run in Figure 3(a). The first two axles crossed the detector in a regular way, i.e. sensor system 2 is affected first, then both systems are affected, afterwards sensor system 1 is affected and finally no sensor system is affected. But for the last three axles the FSM is traversed from state `unaffected` to state `both_systems_affected` via either state `system2 affected` or `system1 affected` and on the same way back (compare to Figure 2). Thus, the internal counters are incremented in an irregular way, because state transitions contrary to the main transition direction occurred. Whether the number of false transitions is within the range of tolerance, is calculated by the equation: $1+a/rl_{max}$, where $rl_{max} \geq 1$ is a parameter, which is defined for every CH type. In the case at hand rl_{max} equals to one and the maximum number of state transitions contrary to the main direction is 6 and $a = 5$. Therefore, a counter value 6 is valid and the five axles are counted correctly as can be seen in Figure 3(b).

If the number of transitions contrary to the main direction would be higher, the corresponding counter would exceed the limit $1 + a/rl_{max}$. This is indicated by setting the counter control token (`countctrl` in Figure 3(a)).

With such test cases the robustness of the counting procedure against interferences and failures can be shown. Additionally, specific scenarios can be simulated to check whether the whole system switches into a safe state if too many interferences occur.

V. Formal Verification

The SystemC model described in Section III is implemented in synthesizeable SystemC. Hence, the design can be synthesized with a frontend, e.g. [5]. The acquired FSM representation of the SystemC design and a specified PSL-property are taken as inputs for a SystemC property checker [4].

In this section we present the formal verification of an aspect of CHs, which is very important for the system to avoid safety hazards. There are several mechanisms to manage failures. One of them is the counter control token (`countctrl`). If the `countctrl` is set, the corresponding TVDS is indicated as occupied. There are two different classes of failures, which have to be considered for setting of the `countctrl`:

1) Erroneous impacts on a single sensor system (including a breakdown of a single sensor system).
2) Erroneous impacts on both sensor systems without axle counting.

The necessary evaluation for these failure classes is carried out in two steps. Firstly, the counters for the two failure classes are calculated. Secondly, the counters are compared to predefined constants. For the impact on a single sensor system the constant is called `s_max` and for the impact on both sensor system it is called `c_max`. For the erroneous impacts on the sensor systems there are also counters, called `s` and `c` for a single axle and `ss` and `cc` for a group of axles. The `countctrl` is set, if $(ss > s\_max) \vee (cc > c\_max)$. This is formulated in a PSL-property and has to hold, universally. This property is verified in 2.7 seconds on a computer with 1GB main memory and an AMD Athlon 64 3500+ CPU running under Linux. But at this point of the verification we still have to prove that the variables `ss` and `cc` have always the correct values, which is performed in the verification steps to follow.

a) Erroneous Impacts on single Sensor Systems In this paragraph the correct value of counter `ss` is proven. Every time the state `unaffected` is reached, the counters for a single axle are added to the counters for a group of axles. Before the counters for a single axle are set to zero, an additional counter is calculated and added to the counter `ss`. To prove the correctness of the latter counter, two steps are needed:

1) The correct implementation of `s` as a function of the state transitions performed during the counting process has to be verified.
2) The correct implementation of `ss` as a function of the sequence of `s` values has to be verified.

The first part is an inductive proof. It takes 7.2 seconds to prove the initial and the induction step. The second part has to hold under any assumption. This means that no inductive reasoning is necessary to prove that the value of `ss` always is the sum of the previous value of `ss` and the currently calculated value of `s`. Thus, we can conclude, that during the evaluation phase `ss` is the sum of all `s`. The proof takes 14.0 seconds.

b) Erroneous Impacts on both Sensor Systems The proof of the correct value of `cc` is analogue to the proof for `ss`. The counter `cc` is the sum of all `c`. And `c` is calculated at the same time as `s` with a different equation. Like `s`, `c` is proven inductively in 6.6 seconds. That `cc` is the sum of all previous `c`, is proven 3.2 seconds.

Altogether, the verification of the counter controller `countctrl` and its invariants took 33.7 seconds.

VI. Conclusions and Future Work

A design flow oriented at efficient modeling and verification of CHs was presented. This design flow is based on SystemC.

The future work consists of the complete formal verification of CHs using bounded model checking and inductive reasoning. We already gave a proof of concept by proving the correct behaviour of the model for the failure class counter control in the preceding section. Another part in future work will be proving completeness of the verification.

References

[1] Accellera. *Property Specification Language Version 1.1*, 2004.
[2] A. Biere, A. Cimatti, E.M. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Design Automation Conf.*, pages 317–320, 1999.
[3] J.R. Burch, E.M. Clarke, K.L. McMillan, and D.L. Dill. Sequential circuit verification using symbolic model checking. In *Design Automation Conf.*, pages 46–51, 1990.
[4] R. Drechsler and D. Große. System level validation using formal techniques. *IEE Proceedings Computer & Digital Techniques, Special Issue on Embedded Microelectronic Systems: Status and Trends*, 152(3):393–406, May 2005.
[5] G. Fey, D. Große, T. Cassens, C. Genz, T. Warode, and R. Drechsler. ParSyC: An Efficient SystemC Parser. In *Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*, pages 148–154, 2004.
[6] T. Grötker, S. Liao, G. Martin, and S. Swan. *System Design with SystemC*. Kluwer Academic Publishers, 2002.
[7] Siemens AG. Az S M Multiple-section Axle Counting System. Copyright, Siemens AG, September 2003.