On the Formal Verification of the FlexRay Communication Protocol

Bo Zhang

On the Formal Verification of the FlexRay Communication Protocol

Bo Zhang[1]

Department of Informatics
Technische Univeisität München
Munich, Germany

Abstract

We present ongoing work on the formal analysis of the FlexRay communication protocol. Isabelle/HOL, a theorem prover for Higher Order Logic, is chosen as our specification and verification system. Essential properties of the FlexRay protocol are identified, formalized and verified. In particular, we show our formal verification of the FlexRay bus guardian component. Furthermore, some insights for the formal verification of the clock synchronization algorithm are exposed.

Keywords: FlexRay, Verification, Bus Guardian, Isabelle/HOL

1 Introduction

FlexRay is an advanced communication system which intends to meet the needs of future in-car control applications, such as those X-by-wire systems. At the core of FlexRay is the FlexRay communication protocol [4]. Specification of the protocol is initiated and maintained by the FlexRay consortium [1]. Currently, the specification of the FlexRay communication protocol is written in plain English together with some SDL(Specification and Description Language) diagrams. Due to the complexity of the specification itself, ambiguity could raise from the semi-formal description. Formal verification techniques, such as theorem proving, can guarantee that crucial properties of the system hold. Since the public release of the FlexRay specification, to our knowledge, there are no known publications on the formal analysis of the protocol. Our aim is to formally specify the protocol and mechanically verify the essential properties of FlexRay.

We choose Isabelle/HOL [9] as our specification and verification system. Isabelle is a generic interactive proof system for implementing logical formalism, and Isabelle/HOL is the specialization of Isabelle for High Order Logic. Isabelle incorporates some *tactics* in order to improve the user's productivity by automating some

[1] Email: zhangb@in.tum.de

parts of the proof process. In particular, Isabelle's *classical reasoner* can perform long chains of reasoning steps to prove formulas. The *simplifier* can reason with and about equations. The tactic *arith* proves linear arithmetic automatically.

Combination of Isabelle/HOL and automatic verification tools, such as SMT(Satisfiability Modulo Theories) solvers, can effectively improve efficiency and the degree of automation as showed in [3,6]. As a long term project, we aim at experimenting this approach for the verification of concrete real world applications, e.g. the clock synchronization in FlexRay.

The rest of the paper is organized as follows: in section 2 we give a brief introduction of the FlexRay communication protocol and our system model for the verification. Section 3 shows our formal verification of the FlexRay bus guardian component. Finally we discuss the conclusion and future work in section 4.

2 The FlexRay Communication Protocol

Critical automotive control systems, e.g. brake-by-wire or steer-by-wire, require a high-bandwidth, fault-tolerant and deterministic communications protocol. It is FlexRay's intention to meet these requirements. The FlexRay provides up to 10Mbit/sec data rate on a single channel. It can also operate on two channels when redundancy is needed. The media access control in FlexRay is based on the repeated communication cycle. The duration of a cycle is fixed after system configuration. The FlexRay communication protocol provides determinism and flexibility by using a combination of time-triggered and event-triggered scheme. Within one communication cycle, the FlexRay protocol offers both a time division multiple access(TDMA) scheme in the static segment and a priority-based minislotting scheme in the optional dynamic segment. The static segment contains a constant number of time slots with the same duration. A number of pre-determined time slots is assigned to each node within one communication cycle. It is only during its reserved time slots that a node is allowed to transmit messages on the bus.

2.1 System Model for FlexRay

The FlexRay system architecture consists of a finite set of computation units, which we call *node*. A node can be either a communication controller or a bus guardian. Each communication controller has a corresponding bus guardian. Nodes communicate by message passing via the underlying communication medium. We assume the reliability of the physical communication medium. Synchronous models are usually preferred for fault-tolerant critical control applications. Our system model for FlexRay is synchronous as we make the following assumptions on the system.

- **Bounded Clock Drift Rate.** For non-faulty clocks there is a known positive constant ρ which bounds the clock time from real time.
- **Bounded Transmission Delay.** The communication delay between two non-faulty nodes is bounded by a constant ϵ.

2.2 Critical Components for Verification

While completely verifying large software systems is almost infeasible, verification of critical components is generally important. We have so far identified two essential components of the FlexRay protocol as candidates for our verification. We give a short description of each component, and the desired properties that we wish to verify.

The Bus Guardian. The bus guardian [5] is used in FlexRay to protect the communication channel against faulty behaviors of communication controllers. Each communication controller has a bus guardian. On the one side, the bus guardian should prevent the communication controller from accessing the communication channel outside its pre-allocated slots. On the other side, the bus guardian should guarantee that messages from non-faulty communication controllers are correctly relayed. We identify four properties of the bus guardian:

- Correct Relay. If a correct communication controller sends a message, its non-faulty bus guardian relays the message.
- Validity. If a non-faulty bus guardian relays a message, then all correct communication controllers receive the message.
- Agreement. If a non-faulty communication controller receives a message, then all non-faulty communication controllers receive the message.
- Integrity. If a message is received by a non-faulty communication controller, the message must have been sent by another non-faulty communication controller.

The Clock Synchronization Algorithm. A time-triggered real time system requires that different nodes have a consistent view of the global time even at the presence of faults. In FlexRay, each node is equipped with its own physical clock. In order to achieve a consistent view of the global time, each node runs an instance of the fault tolerant clock synchronization algorithm. Thanks to the fault-tolerant midpoint algorithm, the FlexRay specification claims that up to two Byzantine faults can be tolerated.

3 Verification of the Bus Guardian Properties

We present the formal verification of *Correct Relay* and *Integrity* in this section. Proof of *Validity* is similar to that of *Correct Relay*, and the proof of *Agreement* shares similarity with that of *Integrity*. Before presenting our verification, we firstly show the notation and assumptions.

We omit the wakeup and startup process. Thus, we assume that communication controllers and bus guardians are on their normal stable state.

Two views of time are involved: Clocktime and realtime. Both range over the real numbers. Lower case letters are used to denote realtime and capital letters are used for Clocktime. Formally, node p's virtual clock or logical clock is defined as a function: $VC_p : realtime \rightarrow Clocktime$. The interpretation is that $VC_p(t)$ is the reading of p's logical clock at real time t. The *bounded clock drift rate* assumption presented in section 2.1 is formalized as follow:

$$\forall t_1, t_2 \in realtime, t_1 < t_2 : (1-\rho)(t_2 - t_1) \leq VC(t_2) - VC(t_1) \leq (1+\rho)(t_2 - t_1).$$

We further assume the correctness of the clock synchronization: the clocks of both communication controllers and bus guardians are closely maintained. Let BG be the set of bus guardians, CC be the set of communication controllers, and δ be the precision of the cluster. We have $\forall p, q \in BG \cup CC, t \in realtime :| VC_p(t) - VC_q(t) | \leq \delta$.

3.1 Verification of Correct Relay

Correct Relay and *Validity* describe the timing relationship between one communication controller and its bus guardian within one slot.

Theorem 3.1 (Correct Relay) *If a correct communication controller sends a message, its non-faulty bus guardian relays the message.*

Informally, the goal of the proof is to show that the relay window of the bus guardian covers the sending window of the sender. Four parameters are introduced as showed in Figure 1. We define a schedule function which denotes the Clocktime of the beginning of a slot: $sched : Slot \rightarrow Clocktime$, where the type *Slot* is defined as natural number. Thus, $sched(n)$ is the Clocktime of the beginning of slot n.

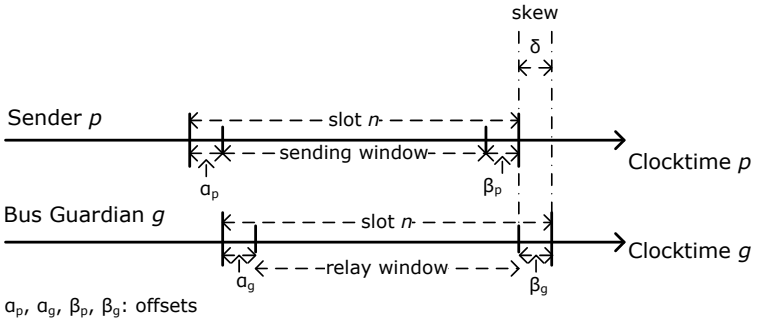

Fig. 1. Timing Information of Sender and Bus Guardian

Proof (Sketch) Suppose after an offset α_p sender p starts the message transmission in slot n. Let t be the real time such that $VC_p(t) = sched(n) + \alpha_p$. Obviously, t is the realtime when sender p begins its sending window of slot n. We require that at realtime $t + \epsilon$ the bus guardian g has been ready to receive the message from sender p within slot n. Therefore, we need to verify that the following formula holds: $VC_g(t+\epsilon) \geq sched(n) + \alpha_g$, where α_g is the offset of slot n for bus guardian g. When ϵ is small it's trivial to prove that $VC_p(t + \epsilon)$ is approximately equal to $VC_p(t) + \epsilon$. For the sake of simplicity, we assume $VC_p(t + \epsilon) = VC_p(t) + \epsilon$. Due to the clock synchronization assumption, the final goal is to show whether the constraint $\alpha_p \geq \alpha_g + \delta - \epsilon$ can be satisfied. The FlexRay bus guardian specification requires $\alpha_p \geq 2\delta$ and $\alpha_g = \delta$. It clearly makes the constraint satisfied.

Suppose $\Delta(n)$ is the message transmission duration in slot n. Let t' be the realtime such that $VC_p(t') = sched(n)+\Delta(n)+\beta_p$. Obviously t' is the latest realtime that the sender closes its sending window. We require that the bus guardian should keep its relay window open until realtime $t'+\epsilon$: $VC_g(t'+\epsilon) \leq sched(n)+\Delta(n)+\beta_g$. According to the clock synchronization assumption and the assumption $\beta_g \geq \beta_p + \delta + \epsilon$, the inequality holds. It should be noted that our verification of the correct

relay is under this assumption, although the FlexRay bus guardian specification does not explicitly state that the condition holds. □

Isabelle's standard simplification tactics can sufficiently handle the equalities and inequalities used in our proof. The Isabelle proof script is showed in the **Appendix**.

3.2 Verification of Integrity

Agreement and *Integrity* characterize the relationship between communication controllers when their bus guardians work properly. Here we detail the proof of the integrity only.

Theorem 3.2 (Integrity) *If a message is received by a non-faulty communication controller, the message must have been sent by another non-faulty communication controller.*

We inductively define a set $\mathcal{C}$ of actions. Actions are of two kinds: *sends* and *receives*. Each message is performed by a node and needs a message. A node is defined as a datatype with two constructors: $CC\ p$ for some communication controller p and $BG\ q$ for some bus guardian q, where p and q are natural numbers. A message is just a freshly introduced type. The set $\mathcal{C}$ of possible communication actions between communication controller and bus guardian is inductively defined by three rules:

- $(sends\ (CC\ p)\ m) \in \mathcal{C} \Longrightarrow (receives\ (BG\ p)\ m) \in \mathcal{C}$
- $(receives\ (BG\ p)\ m) \in \mathcal{C} \Longrightarrow (sends\ (BG\ p)\ m) \in \mathcal{C}$
- $p \neq q \wedge (sends\ (BG\ p)\ m) \in \mathcal{C} \Longrightarrow (receives\ (CC\ q)\ m) \in \mathcal{C}$

With these definitions, The integrity property can be formally defined as:
$(receives(CC\ p)\ m) \in \mathcal{C} \Longrightarrow \exists q, q \neq p \wedge (sends\ (CC\ q)\ m) \in \mathcal{C}$.

The proof follows by rule inversion in Isabelle/HOL. Rule inversion means case analysis on an inductive definition. The proof script is showed in the **Appendix**.

4 Conclusion and Future Work

Isabelle offers good mechanism to formalize communication protocols and algorithms. The verification of the bus guardian properties is a simple case study where the Isabelle/HOL built-in tactics can solve the proofs straightforward with little interaction from the user.

For the verification of the bus guardian properties, we assume the correctness of the FlexRay clock synchronization. The assumption can be discharged if the clock synchronization algorithm indeed works. We are currently studying the formal verification of this algorithm. Our approach is inspired by Schneider's work [10]. Schneider firstly observed that a class of fault-tolerant clock synchronization can be viewed as "result from refining a single paradigm". The paradigm, often called Schneider's abstract clock synchronization algorithm, depends on some general assumptions. Consequently, Shankar [11] mechanically verified Schneider's abstract algorithm with the help of EHDM, a predecessor of the PVS [2] verification system. Shankar reorganized Schneider's theory and gave eleven conditions.

He formally showed if the eleven conditions are satisfied, the clock synchronization holds. Miner's work [8] showed that some of these conditions were too tight or unnecessary.

The FlexRay clock synchronization uses a fault-tolerant midpoint algorithm, which is very similar to Lundelius-Lynch's clock synchronization algorithm [7]. The algorithm itself falls into Schneider's paradigm. However, the FlexRay clock synchronization mechanism uses a combination of both offset correction and rate correction. Informally, offset correction means an adjustment strategy which periodically checks and adjusts the time difference(offset) between clocks in order to bound the maximal offset. Rate correction is another adjustment strategy which periodically checks and adjusts the frequency derivation between two clocks. To our knowledge, Schneider's theory only addresses the offset correction. Currently we are experimenting a way to incorporate the rate correction to Schneider's paradigm.

Acknowledgement

This work has been done during the author's visit of LORIA(Nancy, France) under the supervision of Leonor Prensa Nieto. Stephan Merz has kindly given some helpful tips particularly on the use of Isar proof language.

References

[1] *FlexRay Consortium Homepage*, http://www.flexray.com.

[2] *PVS Specification and Verification System Homepage*, http://pvs.csl.sri.com/.

[3] Barsotti, D., L. Prensa Nieto and A. Tiu, *Verification of Clock Synchronization Algorithms: Experiments on a combination of deductive tools*, in: R. Lazic and R. Nagarajan, editors, *Proceedings of the 5th International Workshop on Automated Verification of Critical Systems (AVoCS 2005)*, ENTCS **145** (2005), pp. 63–78.

[4] FlexRay Consortium, "FlexRay Communication System - Protocol Specification - Version 2.1," (2005).

[5] FlexRay Consortium, "Preliminary Node-Local Bus Guardian Specification - Version 2.0.9," (2005).

[6] Fontaine, P., J.-Y. Marion, S. Merz, L. Prensa Nieto and A. Tiu, *Expressiveness + automation + soundness: Towards combining smt solvers and interactive proof assistants*, in: H. Hermanns and J. Palsberg, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, LNCS **3920** (2006), pp. 167–181.

[7] Lundelius, J. and N. Lynch, *A new fault-tolerant algorithm for clock synchronization*, in: *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing* (1984), pp. 75–88.

[8] Miner, P. S., *Verification of Fault-Tolerant Clock Synchronization Systems*, Technical report, NASA Langley Technical Report Server (1993).

[9] Nipkow, T., L. C. Paulson and M. Wenzel, "Isabelle/HOL — A Proof Assistant for Higher-Order Logic," LNCS **2283**, Springer, 2002.

[10] Schneider, F. B., *Understanding Protocols for Byzantine Clock Synchronization*, Technical report, Cornell University, Ithaca, NY, USA (1987).

[11] Shankar, N., *Mechanical verification of a generalized protocol for byzantine fault tolerant clock synchronization*, in: J. Vytopil, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, Lecture Notes in Computer Science **571** (1992), pp. 217–236.