Towards the Formal Verification of a Distributed Real-Time Automotive System

Erik Endres* Saarland University Germany email@endrese.de

Christian Müller* Saarland University Germany cm@cs.uni-sb.de

Andrey Shadrin* Saarland University Germany shavez@cs.uni-sb.de

Sergey Tverdyshev*† SYSGO AG Germany stv@sysgo.com

Abstract

We present the status of a project which aims at building, formally and pervasively verifying a distributed automotive system. The target system is a gate-level model which consists of several interconnected electronic control units with independent clocks. This model is verified against the specification as seen by a system programmer. The automotive system is implemented on several FPGA boards. The pervasive verification is carried out using combination of interactive theorem proving (Isabelle/HOL) and model checking (LTL).

1 Introduction

There are many works on formal verification of hardware, software, and protocols. However, their interplay in a computer system is not to ignore because even if the hardware and software are correct there is no guarantee that this software is executed correctly on the given hardware. It becomes even more critical when considering distributed embedded systems due to the close interaction of software and hardware parts.

The goal of our project is to show that it is feasible to formally verify a *complex* distributed automotive system in a *pervasive* manner. Pervasive verification [10, 17] attempts to verify systems completely including the interaction of all components, thus minimizing the number of system assumptions. Our desired goal is a "single" top-level theorem which describes the correctness of the whole system.

The subproject Verisoft-Automotive aims at the verification of an automatic emergency call system, eCall [7]. The eCall system is based on a time triggered distributed real-time system which consists of distributed hardware and a distributed operating system.

We use Isabelle/HOL [12], an interactive theorem prover, as the design and verification environment. Our interactive proofs are supported by the model checking technique [21].

Context and Related Work Pervasive verification of a system over several layers of abstraction is introduced in the context of the CLI stack project [2]. However, the application of such verification techniques to an industrial scenario without strong restrictions (e.g. on the programming language) poses a grand challenge problem as by J. S. Moore [10]. Rushby [14] gives an overview of the formal verification of a Time-Triggered Architecture [15] and formally proves the correctness of some key algorithms. Automated correctness proofs for abstract versions of protocols for serial interfaces using k-induction are reported in [13]. There are also recent efforts on the fully automated verification of clock domain crossing issues [9]. It would be highly desirable to reuse results of this nature for a pervasive correctness proof of distributed automotive system. However, putting all these efforts together in a pervasive correctness proof arguing about several layers of abstraction has not been reported.

In the following section we present the automotive system and its components. In Section 3 we describe the hardware environment and system implementation. Section 4 exposes verification challenges. We conclude the paper by summary and future work.

*The authors were supported by the German Federal Ministry of Education and Research (BMBF) in the Verisoft project under grant 01 IS C38

†The reported work has been done while author was affiliated with Saarland University

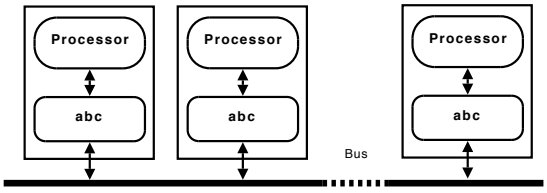

Figure 1: Several ECUs interconnected by a communication bus.

2 Automotive System

The automotive system [8] is inspired by the demands of the automotive industry and based on the FlexRay standard [4]. Our automotive system is a distributed asynchronous communication system represented by a set of electronic control units (ECUs) connected to a single bus. The overview of such a distributed system is illustrated in Figure 1. The ECU is built on the base of a formally verified generic gate-level platform [20]. This platform is a complex gate-level computer system, which consists of a pipelined processor with out-of-order execution and a number of memory mapped I/O devices.

Each ECU has its own clock and contains a bus controller and a processor. The bus controller is attached to the processor via a device interface. Besides the control logic, each bus controller contains two buffers: a send and a receive buffer. We denote the controller "ABC" standing for automotive bus controller. Further we denote by s_i^c the hardware state s in cycle c of the i^{th} bus controller in our network. By $s_i^c.rb$ we denote the content of the receive buffer and by $s_i^c.sb$ the content of the send buffer. Moreover, since we argue about clock domain crossing, we model the translation of digital values to analogous and vice-versa. We use the function $hcy_i : \mathbb{R} \to \mathbb{N}$ to map real time to the corresponding hardware cycles on the ECU i. The function $asr_i(t) : \mathbb{R} \to \{0, 1, \Omega\}$ provides the analogous value of the send register of ECU i in cycle $hcy_i(t)$. Note, that the analogues value gets metastable (Ω) for a short amount of time right after the send register is clocked. During a message transmission we write the content of the send buffer bitwise into this register. We write into it the idle value '1' otherwise. The real time clock period of the i^{th} controller is denoted by τ_i, i.e. one hardware cycle of the ECU i lasts τ_i in the analogous world.

The ECUs communicate in a time-triggered static schedule. These time intervals are the so-called *communication rounds*. A communication round is a periodically recurring time unit which is divided into a fixed number of slots. In each slot, exactly one ECU is allowed to broadcast one message to the bus. Let $\alpha_i(r, s)$ be the point in real time when the slot s of round r is started on the ECU i, and $\omega_i(r, s)$ be the end time of this slot. The start of each round is signalized by one special ECU called *master*, all other ECUs are called *slaves*. As soon as a slave ECU receives this round start signal, it begins with execution of a fixed schedule. Each ECU counts the number of passed slots and, depending on its status in the given slot (sender or receiver), it either samples the bus value or it sends some data to the bus. When a slave ECU reaches the maximal slot number in one round, it goes to an idle state and waits for a new round start signal. The master ECU waits for some predefined amount of time when it is guaranteed that all slave ECUs are waiting for a new round. Only then the master broadcasts a start signal for the new round. The communication protocol as well as the clock synchronization mechanism are described in details in [3].

A system run scenario can be described as follows. Assume the ECU m is acting as a sender in slot s of a round r and before slot s ECU m copied data d from its memory to its send buffer, such that we have at the slot start $s_m^{hcy_m(\alpha_m(r,s))}.sb = d$. After the slot start the ECU m waits *off* cycles before it starts the transmission. The number of cycles *off* has to be big enough s.t. the start time of the slot s on all other ECUs is before $\alpha_m(r, s) + \tau_m \cdot off$. Then, m broadcasts the content of $s_m.sb$ (data d) bitwise to the bus during the next tc cycles (transmission length). At the end of the slot s each receiver ECU contains d in its receive buffer: $\forall i : s_i^{hcy_i(\omega_i(r,s))}.rb = d$.

Such a time-triggered communication requires that all ECUs have roughly the same notion of

time, such that each ECU is aware of the correct slot number and its role in this slot during each message transmission. This is one of the verification challenges (Section 4).

3 Implementation

The verified ECU design has been automatically translated to Verilog [18] directly from formal Isabelle/HOL hardware descriptions and has been synthesized with the Xilinx ISE software. The prototype implementation consists of several FPGA boards which are interconnected into a prototype of a distributed hardware network. Every ECU is running on either Xilinx Spartan-3 and Virtex-2 FPGA development boards [6]. Each board consists of a field programmable gate array (FPGA) and several devices (e.g. LEDs, switches) connected to the I/O-ports of the FPGA chip.Every board has its own clock source, thus, all ECUs are clocked independently. The boards are interconnected via Ethernet cable. The physical layer of the data transmission is tuned to the FlexRay standard and is provided by the low voltage differential signaling driver [11], which generates a differential signal of ± 350 mV. We successfully tested communication between FPGA boards with the help of the hardware logic analyzer Trektronix TLA5204 and the software Chipscope.

4 Verification Challenges

The correctness of the presented distributed system can be split into two parts: local correctness (single ECU) and distributed correctness (asynchronous communicating ECUs).

The local part focuses on the correctness of the processor, the ABC device and their communication, e.g. instructions are correctly executed, the device registers are written and read correctly. More details on the local correctness can be found in [19].

The distributed correctness states that during a run the exchange of data between ECUs is correct, e.g. the sent data of one ECU are the received data on another ECU. Obviously the distributed correctness requires the local one. Moreover, this exchange requires correct asynchronous communication via a FlexRay bus. The state of the bus is a conjunction of outputs of all send registers, i.e. it is an $\wedge$-bus:

$$bus(t) = \bigwedge_{\forall i} asr_i(t)$$

On the receiver side, the bus value $bus(t)$ will be clocked into the analogous receive register, digitalized, and clocked into the receive buffer afterwards. The correctness of the message exchange in the automotive system is based on two properties. First, we have to ensure that if a connection between a sender and receivers is established directly (i.e. we abstract bus by a link), then the low level bit transmission from the sender to all receivers is correct. One of the challenges here is to ensure that the value broadcast on the link is stable long enough so that it can be sampled correctly by the receiver. In our case, if the sender sends n bits, the receiver will sample at least $n - i$ of these bits. The number i is the number of lost bits due to the clock drift between different ECUs. This information loss happens only at the low-level bit transmission. At this level we transmit the message encoded according to the FlexRay standard (each bit is replicated eight times) which defines sufficient redundancy to guarantee the transmission of every single bit of the original information. The correctness of this low-level transmission mechanism cannot be carried out in a conventional, digital, synchronous model. It involves asynchronous and real-time-triggered register models taking into account setup- and hold-times as well as metastability. This part of the pervasive correctness has been formally verified and reported in [16].

The second part is the bus correctness where we have to prove that we can abstract the bus while a sender broadcasts data to the bus. Here, we show that the bus connection can be modeled as a direct link between sender and each receiver. The latter holds only if during each transmission only one sender (one ECU) is broadcasting and all receivers are listening and not sending something (i.e. they are not producing a bus contention). To avoid a bus contention each ECU has to be aware of the correct slot number, i.e. all ABCs have roughly the same notion of the slot start and end times

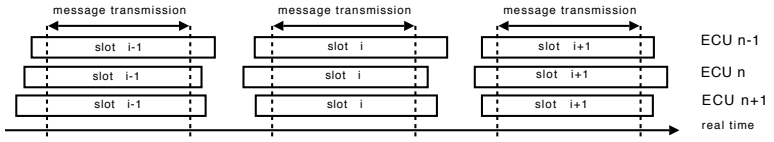

Figure 2: Time notion of an ECU.

(correctness of the scheduling unit). Therefore, due to drifting clocks a synchronization is necessary. We use a simple approach: in the beginning of each round all slave ECUs are waiting for the round start signal broadcast by the master ECU. After this signal all receivers are aware of the current slot number namely zero. All consecutive slots are started and ended locally on each ECU with respect to the maximal clock drift that can occur during a full round. Böhm has formally verified a theorem [3] that if a slave ECU i is waiting and the master ECU m starts a new round r, then the ECU i is aware of each new slot s of this round before the message transmission starts. He also proves that the transmission of each slot ends before the receiver ECU "thinks" that the slot is over. We have significantly extended these theorems and used them as an induction hypothesis to prove that on all ECUs **each** slot of **each** round starts before and ends after the message transmission:

$$\forall \text{ slot } s, \text{ round } r, \text{ ECU } m, \text{ ECU } i \;.\; m \text{ is sender in slot } s \rightarrow$$
$$\alpha_m(r,s) + \mathit{off} \cdot \tau_m > \alpha_i(r,s) \wedge \alpha_m(r,s) + (\mathit{off} + tc) \cdot \tau_m < \omega_i(r,s)$$

Thus, we have shown that each slot of an ECU overlaps with the same slot on all other ECUs during the message transmission as depicted in Figure 2. Since all receivers place '1' on the $\wedge$-bus we show, that during any transmission, the bus contains value of the send register of the sender:

$$\forall \text{ ECU } m, \text{ slot s , round } r \;.\; bus(t) = asr_m(t) \text{ for } t \in [\alpha_m(r,s) + \mathit{off} \cdot \tau_m \;;\; \alpha_m(r,s) + (\mathit{off} + tc) \cdot \tau_m]$$

We prove all real-time properties and complex hardware theorems interactively in Isabelle/HOL. Some properties of hardware with "model-checkable state space" are expressed in LTL and proven automatically [21].

5 Summary

In this paper we presented the status of a pervasive verification of a distributed automotive system. The system is a distributed network of electronic control units interconnected by a single bus involving clock domain crossing. We have successfully built up a working gate-level prototype synthesized from our formal models. We have also partially verified the automotive system. This pervasive verification is very challenging because the system exists on three levels of abstraction: 1. a formal model of an asynchronous *real-time* triggered system on the bus side, 2. a formal gate-level design of *digital* hardware for local properties on the controller side, 3. a formal model as seen by an *assembler* programmer. Moreover, all our models are combined together and are formally specified in Isabelle/HOL theorem prover.

In our previous work we have formally verified a platform for electronic control unit [20], scheduler correctness [3], and low-level bit transmission [16]. As part of the current work we have *consolidated* previous results which was not an easy task due to the combination of results over several layers of abstractions. We are also finishing the verification of the asynchronous message transmission between several ABC devices. The latter includes the verification of the bus correctness (done) and a correct transmission of send and receive messages from the corresponding buffers to / from the bus (in progress).

For future work we see several interesting topics. First, finishing the current work. Then, we can extend the automotive system with fault tolerance, e.g. as sketched in [1]. Another work in progress at our chair is verification of a distributed operating system which runs on top of the presented

distributed system [5]. We also would like to put together formal proofs for the latter operating system and our distributed hardware.

References

[1] Eyad Alkassar, Peter Boehm, and Steffen Knapp. Correctness of a fault-tolerant real-time scheduler algorithm and its h ardware implementation. In *MEMOCODE'2008*, pages 175–186. IEEE Computer Society Press, 2008.

[2] William R. Bevier, Warren A. Hunt, Strother Moore, and William D. Young. An approach to systems verification. *Journal of Automated Reasoning*, 5(4):411–428, 1989.

[3] Peter Böhm. Formal Verification of a Clock Synchronization Method in a Distributed Automotive System. Master's thesis, Saarland University, Saarbrücken, 2007.

[4] FlexRay Consortium. FlexRay – the communication system for advanced automotive control applications. `http://www.flexray.com/`, 2006.

[5] Matthias Daum, Norbert W. Schirmer, and Mareike Schmidt. Implementation correctness of a real-time operating system. In *(SEFM 2009), 23–27 November 2009, Hanoi, Vietnam*, pages 23–32. IEEE, 2009.

[6] Erik Endres. FlexRay ähnliche Kommunikation zwischen FPGA-Boards. Master's thesis, Wissenschaftliche Arbeit, Saarland University, Saarbrücken, 2009.

[7] European Commission (DG Enterprise andDGInformation Society). eSafety forum: Summary report 2003. Technical report, eSafety, March 2003.

[8] Steffen Knapp and Wolfgang Paul. Pervasive verification of distributed real-time systems. In M. Broy, J. Grünbauer, and T. Hoare, editors, *Software System Reliability and Security*, volume 9 of *IOS Press, NATO Security Through Science Series.*, 2007.

[9] Bing Li and Chris Ka-Kei Kwok. Automatic formal verification of clock domain crossing signals. In *ASP-DAC '09*, pages 654–659, Piscataway, NJ, USA, 2009. IEEE Press.

[10] J Strother Moore. A grand challenge proposal for formal methods: A verified stack. In Bernhard K. Aichernig and T. S. E. Maibaum, editors, *10th Anniversary Colloquium of UNU/IIST*, volume 2757 of *LNCS*, pages 161–172. Springer, 2002.

[11] National Semiconductor. *LVDS Owner's Manual*, 2008.

[12] Lawrence C. Paulson. *Isabelle: a generic theorem prover*, volume 828 of *LNCS*. Springer, New York, NY, USA, 1994.

[13] Lee Pike. Modeling time-triggered protocols and verifying their real-time schedules. In *FM-CAD'07*, pages 231–238. IEEE, 2007.

[14] John M. Rushby. An overview of formal verification for the time-triggered architecture. In *FTRTFT '02*, pages 83–106, London, UK, 2002. Springer-Verlag.

[15] C. Scheidler, G. Heine, R. Sasse, E. Fuchs, H. Kopetz, and C. Temple. Time-triggered architecture (tta). *Advances in Information Technologies*, 1997.

[16] J. Schmaltz. A formalization of clock domain crossing and semi-automatic verification of low level clock synchronization hardware. Technical report, Saarland University, 2006.

[17] The Verisoft Consortium. The Verisoft Project. `http://www.verisoft.de/`, 2003.

[18] S. Tverdyshev and A. Shadrin. Formal verification of gate-level computer systems (short paper). In Kristin Yvonne Rozier, editor, *LFM 2008*, NASA Scientific and Technical Information (STI), pages 56–58. NASA, 2008.

[19] Sergey Tverdyshev. *Formal Verification of Gate-Level Computer Systems*. PhD thesis, Saarland University, Saarbrücken, 2009.

[20] Sergey Tverdyshev. A verified platform for a gate-level electronic control unit. In *Formal Methods in Computer Aided Design, FMCAD'09*, IEEE, pages 164–171, 2009.

[21] Sergey Tverdyshev and Eyad Alkassar. Efficient bit-level model reductions for automated hardware verification. In *TIME 2008*, pages 164–172. IEEE, 2008.