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

NASA Formal Methods 2010

Christian Müller

Saarland University, Germany

04/15/2010
### Background

- Verisoft
Background

- Verisoft
- pervasive verification
Background

- Verisoft
- pervasive verification
- automatic emergency call system eCall
Background

- Verisoft
- pervasive verification
- automatic emergency call system eCall
- inspired by FlexRay
Background

- Verisoft
- pervasive verification
- automatic emergency call system eCall
- inspired by FlexRay
- implemented in ML
### Background

- Verisoft
- pervasive verification
- automatic emergency call system eCall
- inspired by FlexRay
- implemented in ML
- Verilog (Shadrin), FPGAs (Endres)
Background

- Verisoft
- pervasive verification
- automatic emergency call system eCall
- inspired by FlexRay
- implemented in ML
- Verilog (Shadrin), FPGAs (Endres)
- electronic control units (ECUs) interconnected by a bus
Automotive Real-Time System
Communication & Clock Synchronization

Each ECU has its local notion of time
- time is split into rounds
- each round consists of $n$ slots

![Diagram showing rounds and slots](image-url)
Automotive Real-Time System

Communication & Clock Synchronization

Each ECU has its local notion of time
- time is split into rounds
- each round consists of $n$ slots

- ECU is a sender or a receiver
- broadcast if *sender*, listen otherwise

![Diagram showing time, rounds, and slots](image-url)
Each ECU has its local notion of time
- time is split into rounds
- each round consists of $n$ slots

- ECU is a sender or a receiver
- broadcast if *sender*, listen otherwise
- all ECUs should be aware of the current slot
Each ECU has its local notion of time
- time is split into rounds
- each round consists of $n$ slots

- ECU is a sender or a receiver
- broadcast if *sender*, listen otherwise
- all ECUs should be aware of the current slot
- synchronization is necessary (clock drift!)
Automotive Real-Time System
Communication & Clock Synchronization

Each ECU has its local notion of time
- time is split into rounds
- each round consists of \( n \) slots

- ECU is a sender or a receiver
- broadcast if *sender*, listen otherwise
- all ECUs should be aware of the current slot
- synchronization is necessary (clock drift!)
Automotive Real-Time System

Bus Controller

∀ real times $t$: $\operatorname{bus}(t) = \bigwedge \forall \operatorname{ECU}_i \operatorname{analogSendRegisterValue}_i(t)$
∀ real times $t : \text{bus}(t) = \bigwedge_{\forall \text{ECU } i} \text{analogSendRegisterValue}_i(t)$
Theorem (Overall Transmission Correctness)

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.
Correctness
Top Level Theorem

Theorem (Overall Transmission Correctness)

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.

Proof Sketch.

1. low lever bit transmission
Correctness
Top Level Theorem

Theorem (Overall Transmission Correctness)

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.

Proof Sketch.

1. low level bit transmission
2. bus correctness (induction on rounds)
Correctness
Top Level Theorem

Theorem (Overall Transmission Correctness)

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.

Proof Sketch.

1. low level bit transmission
2. bus correctness (induction on rounds)
   1. ECUs execute fixed schedule after a round start
Correctness
Top Level Theorem

Theorem (Overall Transmission Correctness)

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.

Proof Sketch.

1. low lever bit transmission
2. bus correctness (induction on rounds)
   1. ECUs execute fixed schedule after a round start
   2. slots overlap
Theorem (Overall Transmission Correctness)

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.

Proof Sketch.

1. low level bit transmission
2. bus correctness (induction on rounds)
   1. ECUs execute fixed schedule after a round start
   2. slots overlap
   3. only senders produce bus activity → no bus contention
Theorem (Overall Transmission Correctness)

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.

Proof Sketch.

1. low lever bit transmission
2. bus correctness (induction on rounds)
   1. ECUs execute fixed schedule after a round start
   2. slots overlap
   3. only senders produce bus activity \(\rightarrow\) no bus contention
   4. after \(n\) slots ECUs are waiting \(\rightarrow\) bus is free
Correctness
Top Level Theorem

Theorem (Overall Transmission Correctness)

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.

Proof Sketch.

1. low level bit transmission
2. bus correctness (induction on rounds)
   1. ECUs execute fixed schedule after a round start
   2. slots overlap
   3. only senders produce bus activity \(\rightarrow\) no bus contention
   4. after \(n\) slots ECUs are waiting \(\rightarrow\) bus is free
   5. master ECU sends a synchronization
Correctness
Top Level Theorem

Theorem (Overall Transmission Correctness)

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.

Proof Sketch.

1. low level bit transmission
2. bus correctness (induction on rounds)
   1. ECUs execute fixed schedule after a round start
   2. slots overlap
   3. only senders produce bus activity $\rightarrow$ no bus contention
   4. after $n$ slots ECUs are waiting $\rightarrow$ bus is free
   5. master ECU sends a synchronization
   6. all ECUs recognize it (by 1) $\rightarrow$ the next round is started
**Theorem (Overall Transmission Correctness)**

At the end of each slot, the receive buffer of all ECUs is equal to the send buffer of the sending ECU at the beginning of that slot.

**Proof Sketch.**

1. low lever bit transmission
2. bus correctness (induction on rounds)
   1. ECUs execute fixed schedule after a round start
   2. slots overlap
   3. only senders produce bus activity → no bus contention
   4. after $n$ slots ECUs are waiting → bus is free
   5. master ECU sends a synchronization
   6. all ECUs recognize it (by 1) → the next round is started
3. message transmission: send buffer - bus - receive buffer (1,2)
Correctness

Previous Results

- Low level bit transmission correctness
  - proven by Schmaltz for two directly linked 1-bit registers with different clocks
  - receiver samples $n$ of $m$ sent bits, $n \leq m$
Correctness

Previous Results

- **Low level bit transmission correctness**
  - proven by Schmaltz for two directly linked 1-bit registers with different clocks
  - receiver samples $n$ of $m$ sent bits, $n \leq m$

- **Scheduler Correctness**
  - proven by Boehm for three controllers (linked to master only)
  - after synchronization – no slot boundaries within transmission
Correctness

Our Progress

- computation model of $n$ ECUs
## Correctness

### Our Progress

- computation model of $n$ ECUs
- interconnection by a bus
Correctness
Our Progress

- computation model of $n$ ECUs
- interconnection by a bus
- proof of the initialization routine
Correctness

Our Progress

- computation model of $n$ ECUs
- interconnection by a bus
- proof of the initialization routine
- used previous results to show the bus correctness*
Correctness

Our Progress

- computation model of $n$ ECUs
- interconnection by a bus
- proof of the initialization routine
- used previous results to show the bus correctness*
- future work
  - generalization of the bus architecture
Correctness
Our Progress

- computation model of $n$ ECUs
- interconnection by a bus
- proof of the initialization routine
- used previous results to show the bus correctness*
- future work
  - generalization of the bus architecture

message transmission
Challenges

- all proofs are done with Isabelle + NuSMV (Tverdyshev)
Challenges

- all proofs are done with Isabelle + NuSMV (Tverdyshev)
- integration / combination of proofs
Challenges

- all proofs are done with Isabelle + NuSMV (Tverdyshev)
- integration / combination of proofs
  - Low Level Bit Transmission Correctness

- Scheduler Correctness
Challenges

- all proofs are done with Isabelle + NuSMV (Tverdyshev)
- integration / combination of proofs
  - Low Level Bit Transmission Correctness
    - too strong assumptions (e.g. unnecessary $\forall$s)
  - Scheduler Correctness
Challenges

- all proofs are done with Isabelle + NuSMV (Tverdyshev)
- integration / combination of proofs
  - Low Level Bit Transmission Correctness
    - too strong assumptions (e.g. unnecessary ∀s)
    - inconsistent assumptions (e.g. unbound variables)
  - Scheduler Correctness
Challenges

- all proofs are done with Isabelle + NuSMV (Tverdyshev)
- integration / combination of proofs
  - Low Level Bit Transmission Correctness
    - too strong assumptions (e.g. unnecessary ∀s)
    - inconsistent assumptions (e.g. unbound variables)
  - Scheduler Correctness
    - semantics transformations (e.g., initialization)
Challenges

- all proofs are done with Isabelle + NuSMV (Tverdyshev)
- integration / combination of proofs
  - Low Level Bit Transmission Correctness
    - too strong assumptions (e.g. unnecessary $\forall$s)
    - inconsistent assumptions (e.g. unbound variables)
  - Scheduler Correctness
    - semantics transformations (e.g., initialization)
- a complete formalization and implementation of the entire model before proofs would be VERY helpful!
Thank you!

Questions?