High-Confidence Bus Architectures: The Backbone of Automotive Cyber-Physical Systems

Lee Pike[*]
Galois, Inc.[†]

February 27, 2008

Introduction and Scope Automotive cyber-physical systems (CPS) encompass nearly every research challenge offered by high-confidence computing. To scope this position paper, I will focus on open research questions in the design and assurance of fault-tolerant real-time automotive communication buses. An example of a fault-tolerant automotive bus in development today is FlexRay^{TM}, and there are a variety of fault-tolerant buses being researched and developed for avionics systems [4]. Such buses provide the intra-vehicle communications network for the most safety-critical applications, such as drive-by-wire, brake-by-wire, and throttle-by-wire systems. That said, I consider *bus architectures* in this paper broadly: this includes the buses themselves, bus interface units, and the interacting processing units driving the sensors, actuators, and other CPS applications.

In the following, I describe three broad research agendas I believe to be paramount to the success of high-confidence CPS systems. The first of these is a community effort to specify and build an open bus architecture to act as a springboard for future research efforts. The second describes research challenges in *formally* specifying and verifying bus architectures for automotive CPSes. The safety-critical and security-critical nature of these systems, coupled with their complexity and multiple layers of abstraction, suggest that mathematically-rigorous specification and verification is necessary to have confidence in their correctness. Finally, I describe the research challenges in building bus architectures that at once integrate applications while providing needed partitioning.

A Open High-Assurance Automotive Bus Architecture A variety of open problems exist in the design of a bus architecture suitable for next-generation automotive CPSes. To spur research in this area, we need an open-source bus architecture that is suitable for use in education, academic research, and industrial research and development. The availability of such an architecture would provide a basis for researchers to make rapid advances and to explore design trade-offs for specific aspects without having to design an entire communications infrastructure from the ground-up. It would aid in concretizing research ideas in fault-tolerance, real-time systems, distributed systems, control systems, etc.

To serve the needs of the community, the architecture should be highly-parameterizable on the following dimensions: the extent and kind of fault-tolerance it provides, the number of nodes and interconnects supported, middleware and operating-system requirements, transmission medium requirements, (e.g., fiber optic, coaxial cable, etc.), and node requirements (e.g., FPGAs, ASICs, co-processors, etc.). Ideally, a variety of open-source implementations derived from the open design would be made available.

[*]Email: `leepike@galois.com`; personal webpage: `http://www.cs.indiana.edu/~lepike`; phone: +1 503 626 6616 ext. 135.

[†]Address: 12725 SW Millikan Way, Suite 290, Beaverton, OR 97005, USA; webpage: `www.galois.com`.

Some progress has already been made in the related area of high-confidence avionic buses. The Scalable Processor-Independent Design for Enhanced Reliability (SPIDER) is a fault-tolerant real-time bus designed and developmented at the NASA Langley Research Center with a publicly-available design and open-source VHDL code for one implementation [5].[1] SPIDER is a platform built from a novel ultra-fault-tolerant design to research the use of formal verification in the design and development of bus architectures. Its design may not be suitable for automotive research since, for example, its fault-tolerance is at the cost of additional hardware and mandated time-triggered behavior [1]. However, lessons can be learned from the design of SPIDER (and avionics buses, in general), and we should be inspired to have a similarly open platform for research.

Cross Abstraction-Layer Specification and Verification An exciting research challenge offered by automotive CPSes is to overcome the following paradox: the safety-critical and security-critical nature of automotive CPSes dictates the need for mathematical certainty of correctness. This need is exacerbated by their complexity, and it is this very complexity that makes formal analysis difficult. Much of the complexity arises from the multitude of abstraction layers; thus, a research challenge is to come up with new models and verification techniques that cut across and map between traditional abstraction layers. To give a more concrete idea about these layers and their challenges, I give two examples below:

- *From distributed protocols to distributed code*: Beyond a few research efforts, the current state-of-the-art in protocol verification is to verify the distributed fault-tolerant protocols at the algorithmic level. However, these protocols are implemented (say on FPGAs) as individual communicating state-machines. Furthermore, the hardware implementations may compose or pipeline protocols, perform other computations concurrently, and deal with implementation-specific timing and data issues. The upshot is little assurance that (the formally-verified) algorithmic specifications are implemented correctly, thereby partially undermining the increased assurance gained from the algorithmic verification. Research to refine formally distributed fault-tolerant protocols into distributed state-machines is needed, including theoretical breakthroughs as well as tools, such as compilers and domain-specific languages. Indeed, I imagine fertile grounds for collaboration with seemingly-unrelated research, such as the current interests in multi-core computation.
- *From real-time constraints to hardware schedules*: The physical world imposes constraints and uncertainties on an implementation that make it difficult to meet hard real-time constraints. These uncertainties include, for example, clock jitter and skew, wire delays, signal settling error, sensor latency, and so forth. Despite these uncertainties, we wish to guarantee hard real-time performance. New models and techniques for proving timing bounds are met is needed. Realistic timing characteristics often involve many tedious calculations on real-time and clock-time variables; recently-developed techniques in decision procedures (also known as *satisfiability modulo theories*) may lead to higher confidence that timing constraints have been met [3]. One upshot is that with *provable* bounds, "fudge-factors" may be eliminated, yielding better throughput.

Integrated yet separated applications Two desiderata of automotive bus architectures initially seem to be at odds. On the one hand, bus architectures are supposed to *integrate* applications—for example, a brake-by-wire and steer-by-wire system execute on the same processing unit and communicate over the same bus. The integration has numerous advantages including providing "off-the-shelf" fault-tolerance and hard real-time guarantees for the applications.

On the other hand, a bus architecture must also *separate* applications—two separate applica-

[1]Publications relating to SPIDER can be found at `http://shemesh.larc.nasa.gov/fm/spider/` (formal specifications and proofs of its protocols are also available here). Open-source VHDL code for SPIDER's bus is available, too: `http://opensource.arc.nasa.gov/software/robus-2/`.

tions should not interfere with one another. This is particularly relevant in automotive systems that may serve applications with a broad range of safety and security criticality. For example, some applications may introduce risks due to accepting, for example, extra-bus communication mechanisms, like Bluetooth®. My steer-by-wire system should not be susceptible to Bluetooth attacks!

Real-time partitioning operating systems can provide high-assurance separation between applications. However, we do not always wish to have complete time and space partitioning; rather, we want to have controlled communication. For example, a brake-by-wire system may deliver real-time data to a diagnostics system, but we may want to ensure information flow occurs only in one direction (from the braking system to the diagnostics system) if the diagnostics system is of lower-confidence. Possible solutions may be borrowed from the security world: for example, a high-assurance multilevel file-server (such as Galois's Trusted Service Engine [2]) on a partitioning operating system may be sufficient. How to balance separation and integration requirements within physical constraints (e.g., weight, size, performance) is an open problem.

Conclusions I have outlined three broad research agendas I believe to be essential to next-generation automotive CPS systems. While a high-confidence bus architecture is only one component in a full high-confidence automotive CPS, it is an essential one, and I believe answering early the research questions outlined above will naturally shed light onto other open problems in automotive cyber-physical systems.

Author's Biography

Since 2005, Dr. Lee Pike is a Research Engineer at Galois, Inc., the mission of which is to create trustworthiness in critical systems by transitioning innovative research into everyday practice. At Galois, Dr. Pike has led multiple million-dollar research projects in the areas of security, formal verification, and virtualization.

From 2003 to 2005, Dr. Pike was a staff scientist in the Formal Methods Group of the NASA Langley Research Center. He researched the design and formal verification of fault-tolerant real-time bus architectures. In 2003, Dr. Pike was also a visiting researcher at the National Institute of Aeronautics.

Dr. Pike holds a Ph.D in Computer Science from Indiana University, Bloomington (2006) and has published in the areas of formal methods and embedded systems. He recently won a Best Paper award at the IEEE FMCAD conference [3]. A full list of publications can be found on Dr. Pike's personal webpage, `http://www.cs.indiana.edu/~lepike`.

References

[1] H. Kopetz. *Real-Time Systems.* Kluwer Academic Publishers, 1997.

[2] D. McNamee, S. Heller, and D. Huff. Building multilevel secure web services-based components for the global information grid. *CrossTalk - The Journal of Defense Software Engineering*, May 2006.

[3] L. Pike. Modeling time-triggered protocols and verifying their real-time schedules. In *Proceedings of Formal Methods in Computer Aided Design (FMCAD'07)*, pages 231–238. IEEE, 2007. Available at `http://www.cs.indiana.edu/~lepike/pub_pages/fmcad.html`. Best Paper Award.

[4] J. Rushby. Bus architectures for safety-critical embedded systems. In *EMSOFT 2001: Proceedings of the First Workshop on Embedded Software*, pages 306–323. Springer, 2001.

[5] W. Torres-Pomales, M. R. Malekpour, and P. Miner. ROBUS-2: A fault-tolerant broadcast communication system. Technical Report NASA/TM-2005-213540, NASA Langley Research Center, 2005.