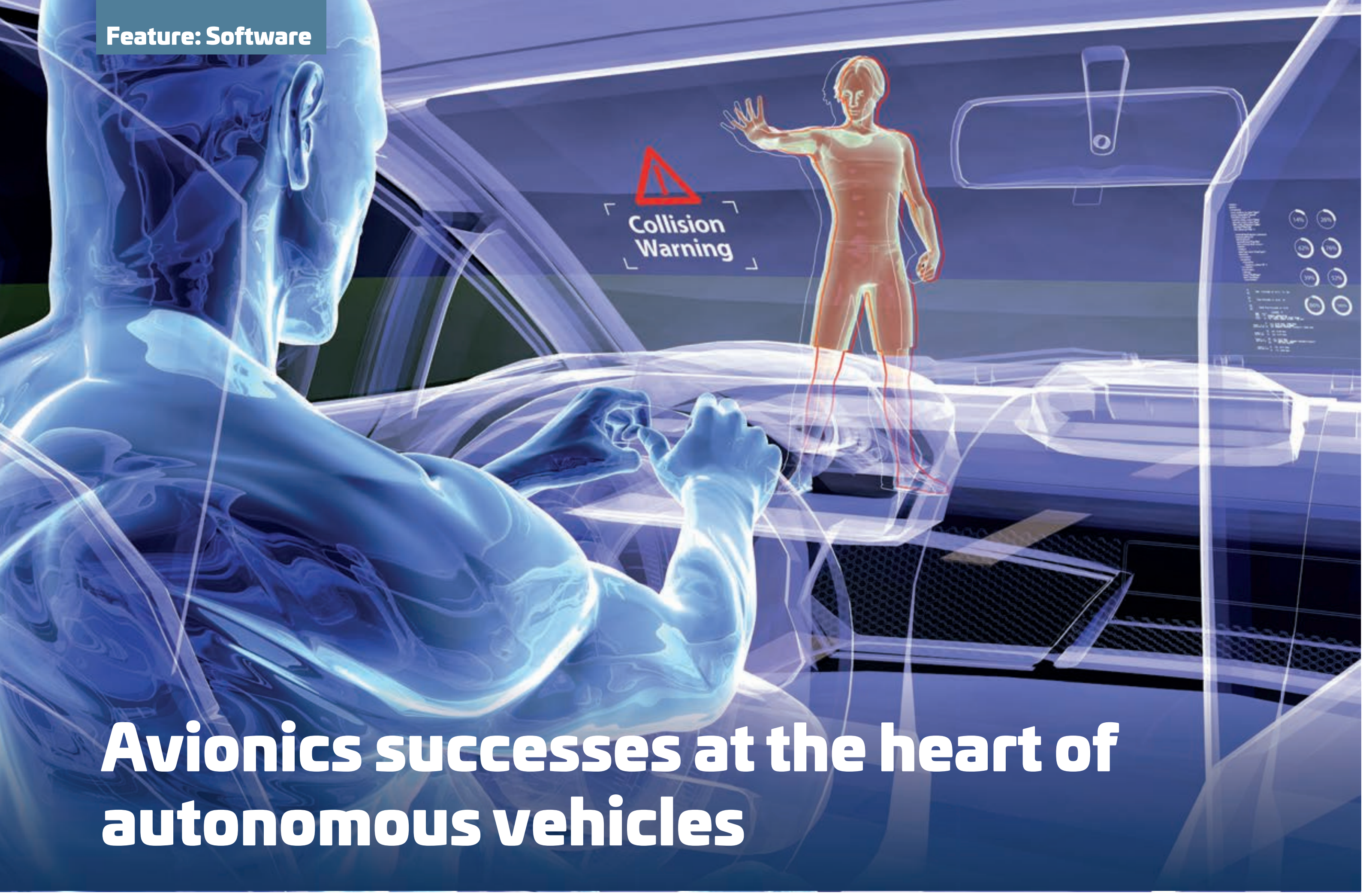

Avionics successes at the heart of autonomous vehicles

By Ian Ferguson, Vice President, Marketing and Strategic Alliances, Lynx Software

We are seeing exciting and innovative work in autonomous cars, especially around Level 4, where the car is in complete control of certain parts of the driving. It is helpful to think about convergence between automotive systems and avionics: autonomous vehicles can combine the extraordinarily successful safety culture of aviation, where even one accident is unacceptable, and the extraordinary flexibility and affordability of the automobile.

System architecture

Achieving this flexibility demands a major focus on driving down costs, vehicle weight and the volume of electronics on board cars. After the engine, the electronics and wiring harness are the most expensive and heaviest parts of a vehicle.

System architectures are being re-evaluated to deliver a tenfold reduction of cabling lengths, from the current 1.5+ kilometres. Instead of the traditional approach of creating different domains for various data networking protocols, we are seeing zonal architectures, where high-performance controllers manage many functions in sections of the vehicle. Hence, we are seeing:

a) Ethernet to connect subsystems inside the vehicle – some are even evaluating 10Gb technology here;
b) A consolidation of processing functionality into fewer, very high-performance electric control units (ECUs).

The range of networks inside the car includes the controller area network (CAN) – which takes care of the powertrain and related functions; local interconnect network (LIN) – for passenger/driver comfort functions like climate control, lighting, seat adjustment; media-oriented system transport (MOST), developed for infotainment; and FlexRay for anti-lock braking (ABS), electronic power steering (EPS) and vehicle stability functions. Each network must be highly secure, to reduce the "attack surface" for potential hackers. Attack surface of a software environment is the sum of different points where a hacker could extract or insert data. For example, simple sensors that send encrypted information to a centralised node need their security improved locally and at the central point. It is obvious that these nodes are processing traffic with very different response requirements.

The challenge is how to implement this in a vehicle where certain systems are mission-critical and need addressing in microseconds. The self-driving vehicles on the road today are effectively servers on wheels. Intel Xeon + nVidia platforms deployed in prototypes will most likely be ousted in favour of solutions that fit significantly smaller footprints, costs and power envelopes – and that race is on between companies!

The attack surface of a software environment is the sum of the different points where an unauthorised user might try to insert or extract data

The processing system

Autonomous vehicles, just like aircraft, demand secure software platforms, which must be achieved without the massive overheads encountered in avionics. We see multicore heterogeneous processors that feature general-purpose processors, but also potentially GPGPUs, programmable logic, more specialist real-time cores hardware accelerators, and more. From a software perspective, there's need to combine rich operating systems (like Linux) on which a wide set of applications can be used almost immediately, with a guaranteed real-time determinism for certain functions.

The hypervisor layer will need to simultaneously host safety-critical applications up to ISO 26262 ASIL D, support non-real-time OSs (such as Android and Linux), AUTOSAR (the AUTomotive Open Systems ARchitecture) kernels, and bare-metal applications.

In several current systems, diverse functionality makes them look heterogeneous, but, in reality, they are separate processing systems running different software. The shift to allocating processing of different tasks more dynamically, coupled with the industry's desire to reduce lock-in to a specific vendor means that these systems are increasingly using hypervisor technology and executing different operating systems and applications on top of them.

Separation kernel needed

Virtualised embedded software architecture has traditionally placed much of the burden on the hypervisor and/or operating system, which creates platform dependencies, impacting performance (due to extra copies and context switches) and raising challenges due to:

- Shared address space;
- Shared CPU privileges;
- Common arbitration points;
- Global resource pools;
- Compounding code branches;
- Compounding control flow timing;
- Large co-dependent code base to certify.

Now this complexity has changed to simplicity, with the hypervisor working seamlessly in the background.

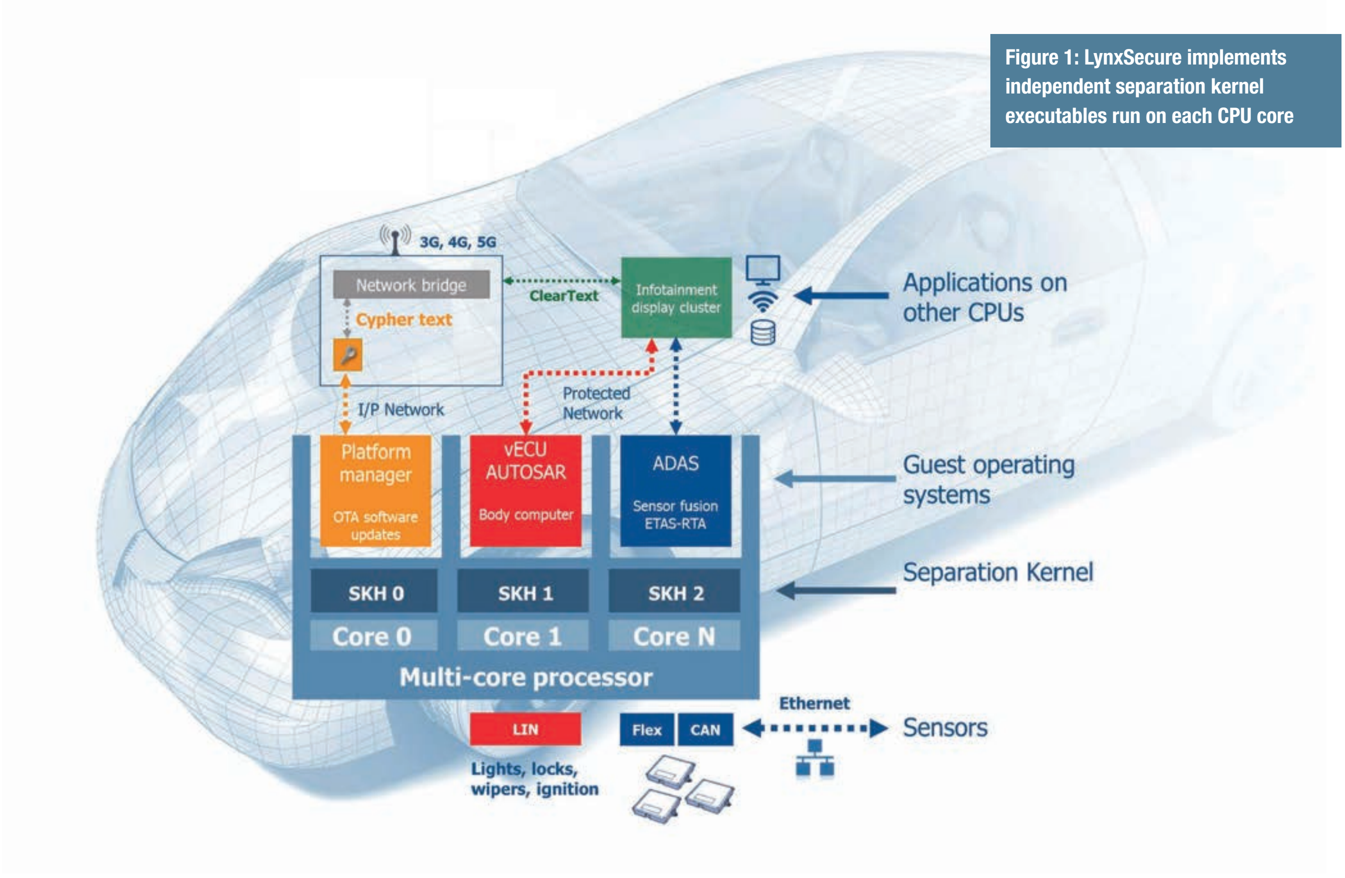

Figure 1: LynxSecure implements independent separation kernel executables run on each CPU core

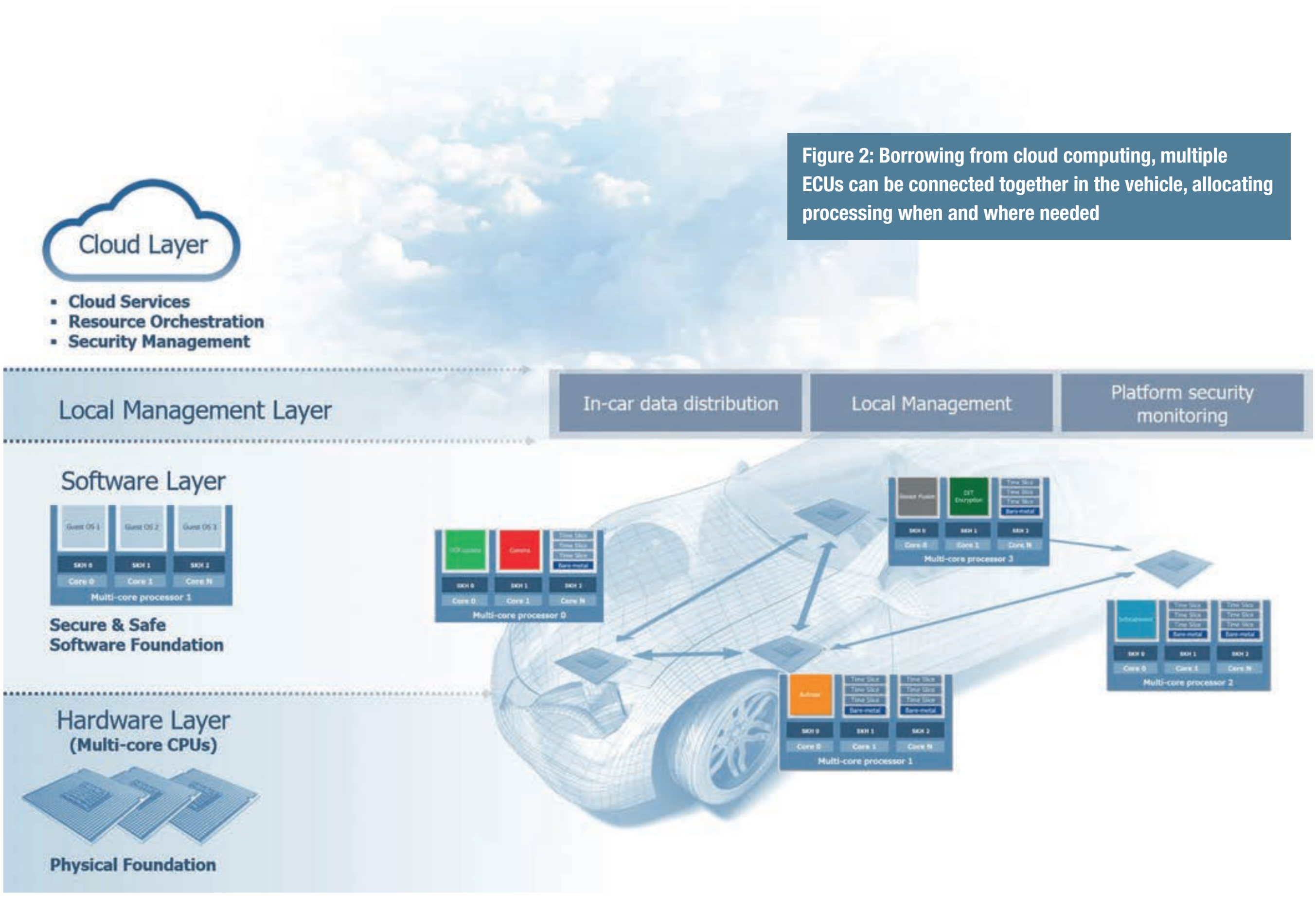

Figure 2: Borrowing from cloud computing, multiple ECUs can be connected together in the vehicle, allocating processing when and where needed

LynxSecure implements independent separation kernel executables run on each CPU core, which partitions platform resources into isolated virtual machines (Figure 1), with additional functionality layered on with "subjects" and "guests". Each additional layer is constrained to a specific virtual machine definition. The separation kernel precisely defines the virtual machine for each guest, such as hardware rights and privileges, communications paths and hypercall permissions. Engineers define their own systems; there's no Master, Trusted, Root, Helper, or Service RTOS. There is no post-boot modification of memory or other resources assigned to virtual machines, hence no single point of failure.

Many markets oscillate between distributed and centralised computing. This push will minimise the cost of sensors and localise more of the processing.

Cost and power needs can be driven down even further. Many systems require just minimal processing most of the time. Borrowing from cloud computing, what if these "systems of systems" – i.e. multiple ECUs connected together in the vehicle – could allocate processing when needed; see Figure 2?

Instead of the traditional approach of creating different domains for various data-networking protocols, we are seeing zonal architectures

Standards compliance

Compliance with relevant safety standards like ISO26262 – and increasingly ISO21434 – is extremely important, whether creating traditional physical automotive components, or, in Lynx's case, virtual ones like hypervisors. We've learnt from avionics that it is critical to write detailed requirements documents, with detailed traceability all the way to the fine-grained hardware functionality, which in turn makes verification easier. With spiralling sizes of code bases in the car, the only viable way is to compartmentalise code. Breaking down the code into manageable chunks, issues and source code is key, proving that the operation of a code base is isolated from others and then demonstrating the requirements are met by tracing them to other artefacts, including tests.

In our example, use of a full AUTOSAR-compliant runtime platform, in this case from ETAS, consisting of RTA-OS (operating system for deeply-embedded ECUs), RTA-RTE (AUTOSAR Runtime Environment generator) and RTA-BSW (AUTOSAR-compliant Basic Software). Existing AUTOSAR software can be integrated into a powerful ECU whilst providing the safety, security and freedom from interference required by the most demanding applications. EW