Safety and Security Aware Pre-silicon Concurrent Software Development and Verification

Frank Schirrmeister, Joe Fabbre, Max Hinson

Cadence Design Systems Inc., Green Hills Software LLC

Abstract

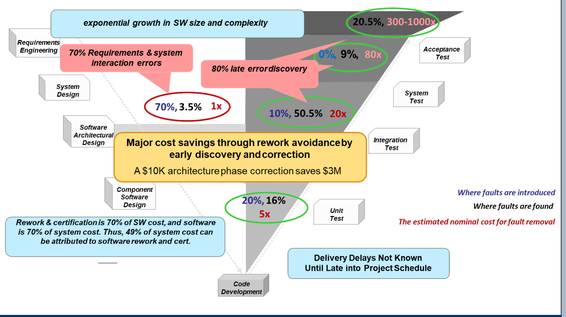
Simultaneously designing and testing both software and SoC designs before silicon is an enticing goal to reduce the time and cost of building embedded devices. The earlier that applications can run on a reliable representation of the SoC, the better. Merging the software and SoC paths before silicon saves time and money and creates a virtuous cycle that gets the embedded product to market faster.

This paper describes a new pre-silicon continuum for concurrent SoC and software development and verification for safety and security critical systems, composed of a safety-certified RTOS and advanced C/C++ development tools that continuously support SoC verification from the earliest functional simulator (virtual prototype with Arm Fast Models) through RTL emulation and FPGA prototype stages on the path to first silicon.

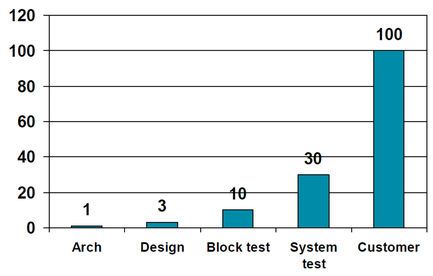
The continuous convergence brings more safety, security and verified reliability while establishing a more mature software enablement foundation for lead customers and partners at the time of first silicon.

Cost of Software Bugs

Studies have shown that the relative cost to fix an error highly depends on the phase in which it is found. Compared to the requirements phase, the cost can be 3-6 times as high in the design phase, 10 times in the coding phase, 15-40 times in the development and testing phase, 30-70 times in acceptance testing, and 40-1000 times in the operational phase.



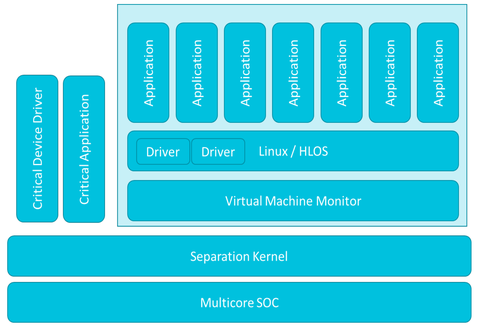
Cost of Hardware Bugs



Safety and Security Considerations for Software

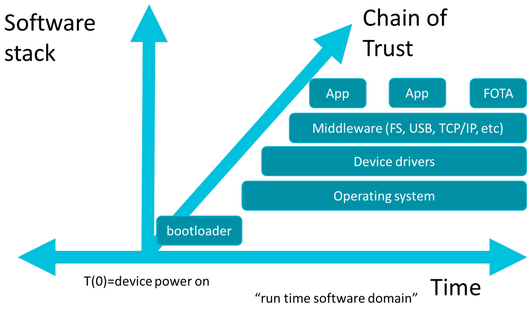
The most effective approach to developing a safe and secure system begins with decomposing the system into separate components. Each component is analyzed for its size, complexity, resource needs, and safety and security requirements. The components that will be certified for safety or security must be designed to be very small, simple, and isolated from other components in the system. In turn, these components will be much simpler to test and certify. Non-critical components, on the other hand, may be large and complex, as they will not need to be certified.

Proper separation of the safety or security-critical components from the non-critical components is critical. Without this isolation, the system design can be compromised. A Separation Kernel guarantees resource isolation between application-level programs, allowing a system architect to separate critical device drivers and applications from applications that may be less safe or secure. Additionally, a Separation Kernel provides the isolation required to safely and securely run guest operating systems alongside critical real-time applications.



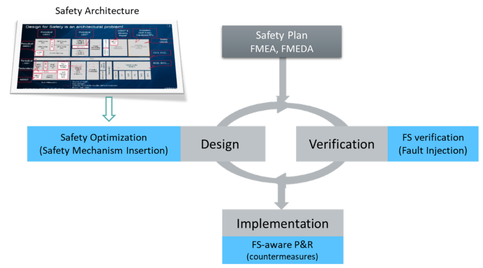
Chain of Trust in a Typical Software Stack

Security is multi-faceted and must address more than just architectural design. Functions such as secure software boot, key storage and infrastructure, software update systems, and cryptography for data at rest and in transit are critical parts of a safe and secure system. Figure 3 shows how the chain of trust is built across the software stack from bootloader, through operating systems, device drivers, middleware and user applications.



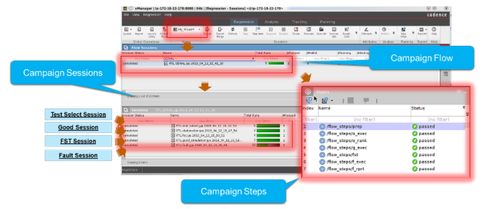
Safety and Security Considerations for Hardware

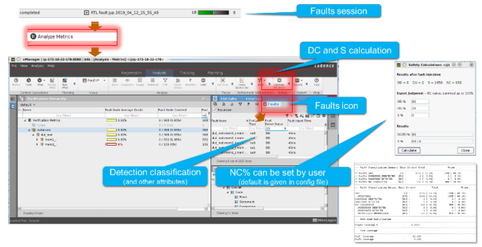
For hardware, safety and security have to be linked to the traditional design/verification and implementation flows, to include safety mechanisms and meet the hardware metrics according to ASIL standards. Safety metrics, Power Performance Area (PPA), verification time and automation have to be considered.



Fault Campaign Management Execution and Analysis

From a central cockpit, the flow of campaigns is controlled, campaign sessions and steps are executed. The result of fault session is collected from actual execution in dynamic engines like simulation and emulation, and accumulated, illustrating to the user the various attributes like classification of a detection.





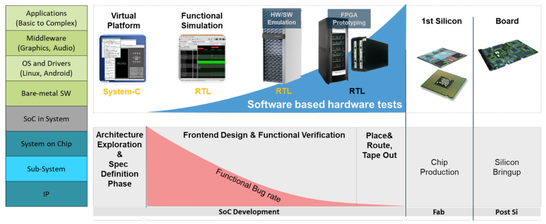
There are many opportunities to optimize for safety, security, and performance when software and hardware designers work together early in the process of system design. There are some functions that may be performed in hardware, rather than software, such as lock step cores, ECC memory, Spectre mitigations, and built-in self test (BIST).

During the development phase, this reduces the amount of software that needs to be designed, written, tested, and certified, and reduces the number of bugs. It also reduces the software's need for hardware resources.

Shift Left with Early Hardware-Software Integration

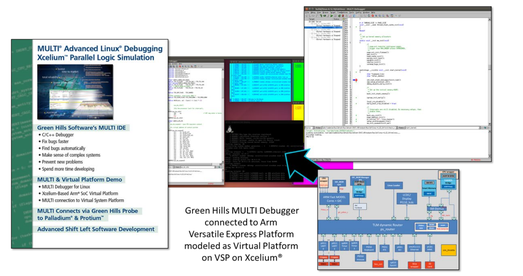
Early in the design flow, virtual prototypes at the transaction level can execute the same software that later is loaded on the board using register accurate descriptions in SystemC.

At the register transfer level (RTL), simulation, emulation and prototyping provide different trade-offs between execution performance, debug insight, accuracy and bring-up time (i.e. the time to when the execution model runs functionally correctly). Simulation and emulation are initially focused on hardware verification, with prototyping focused on software development. Once chips are back from production, software development can proceed on the actual silicon.



Results

An Arm-based virtual platform using a Cadence Virtual System Platform based on Xcelium SystemC simulation was developed. The virtual platform is executing an OS – in this case Linux – for early software bring-up. The virtual platform is connected to the Green Hills Multi IDE and debugger for software development.



Software teams use production-grade C/C++ development tools and RTOS on verification engines prior to silicon availability.

The entire software schedule can experience a "shift-left", where pre-silicon development engines, virtual prototyping, RTL simulation, emulation and FPGA-based prototyping can be used before first silicon to develop and debug software applications, middleware, RTOS and device drivers. They even enable more developers to simultaneously develop/test code after first silicon, as early-revision boards are often very scarce.

Safety certified RTOS and C/C++ compilers allow system designers to take into account safety and security from the very beginning of the design process, allowing more complete software enablement on first silicon once it is available. Processor manufacturers can launch first silicon pre-tested devices with production-grade RTOSs.