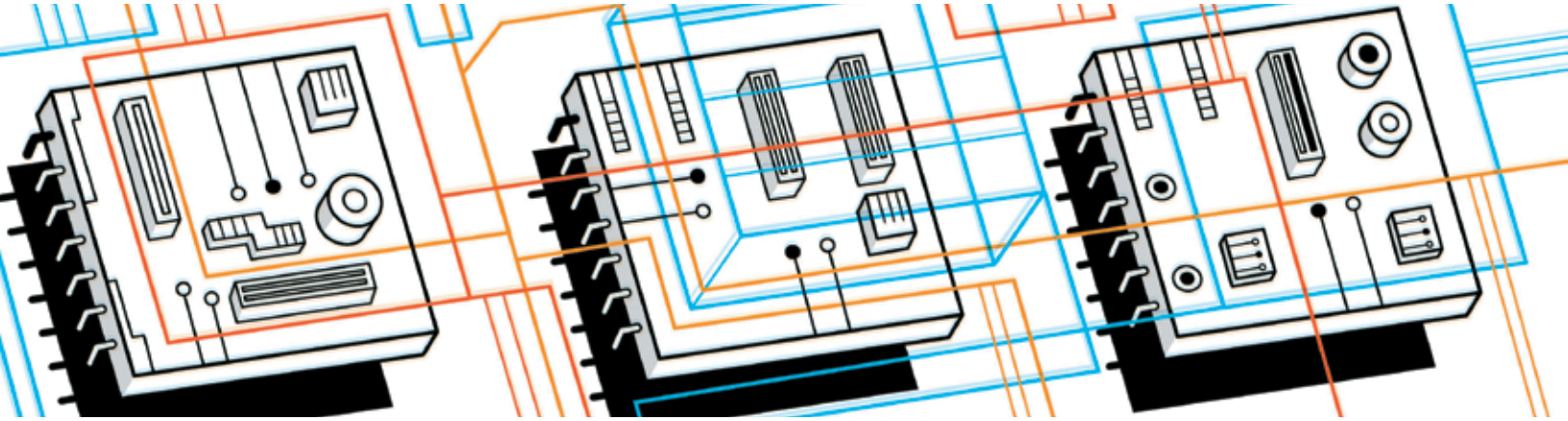


Harry Campbell

When software meets hardware: Excellence in embedded-software development

Embedded software has become essential to the success of most types of new semiconductors. Yet some semiconductor companies still resist the idea that they are selling not just hardware but also, increasingly, software. A blueprint can help in better integrating them in your organization.

Harald Bauer and Ondrej Burkacky

For many years, software was an afterthought for semiconductor companies. When software did get attention, it was limited mostly to basic firmware operating the integrated circuits (ICs) that the companies produced. But in the last five years, as hardware has become increasingly commoditized and customers demand shorter time to market, the importance of embedded software has grown.

At one time, hardware designers were the dominant class of engineers in most semiconductor R&D organizations. Now, given the rise of mobile devices, most IC designers employ more software developers than hardware engineers. In consumer-facing markets, that evolution has come quickly. Through work in the wireless-handset sector, it was observed that more than 60 percent of engineers are engaged in software development or testing, compared with roughly 40 percent three years ago and less than 20 percent in 2008.

Companies undergoing the transformation from hardware- to software-centric business models typically find that several aspects of their existing processes lead to productivity losses, quality problems, rework due to late defect detection, and budget overruns. These include lack of modularization, manual testing regimens, and hardware-led development processes that do not fit the agile-development model required

for software. Several ways to overcome these challenges have been identified, but the three discussed below usually have the most impact.

Giving software its own driver's seat

Because their historical operations were hardware-centric, semiconductor companies' supporting structures remain that way. Hardware timelines drive both overall company planning cycles and the operations of embedded-software divisions. This approach is not well matched to the agile-development methodology common in software development. (Software releases tend to be in ranges of hours or days, whereas new hardware typically takes months to develop.) Instead of software development coming along for the ride with hardware development, these activities should be placed on separate but coordinated tracks, with frequent release cycles. To achieve this, project clocks should be aligned to an overarching system plan, featuring smooth integration and timely definition of requirements on both the hardware and software sides of the development team. Parallel development, with frequent release cycles, should be the desired end state.

Overcoming practical obstacles

From a system-architecture standpoint, it may seem difficult to place embedded software on a different development track than hardware. Certain portions of the software, such as firmware, should be closely linked to the hardware. The use of abstraction layers, however, can help to decouple software stacks and allow for internal optimization of these modules' interfaces and communications protocols. This decoupling approach can also make it easier to migrate software stacks to new hardware, thereby fostering reuse and cutting down on the need for rework.

Release cycles can be automated using a software-development tool chain[1] that handles automatic release management with multiple modules and ideally includes the "virtual prototype" of the target hardware for verification purposes. Several players in the embedded-software field have shown that variable release cycles of as little as three hours to one week are feasible, gaining significant flexibility, reducing bug-fixing effort, and shortening the overall project timeline.

Integrating verification processes

The verification process should transition from a rigid hardware focus to one that has an integrated development tool chain with a fully automated verification work flow. Testing should be continuous, and a priority should be finding bugs early and fixing them before they get integrated on the system level. Continuous testing can be made possible by virtualizing the entire system stack (for example, in wireless, including the base station, "air" interface, mobile antenna, mobile-software stack, and baseband chip) and then

[1] In this article, software tool chains are referred to in their purest sense—that is, a set of programming tools with logical, sequential relationships—rather than the common usage that refers to any collection of programming tools.

Software and hardware development should be placed on separate but coordinated tracks, with frequent release cycles.

conducting automated testing of the virtual stack at "precommit" (when the developer submits a final change request to be included in the system). A reduction of the defect density by more than 50 percent is feasible with this approach.

Seeing the impact

Programs employing these levers can generate significant impact. Several companies have improved time to market by 30 to 40 percent, while product-quality scores rose by up to 50 percent and overall productivity increased by roughly 30 percent (exhibit).

Such a transformation can take more than 18 months, but initial results from some initiatives can deliver measurable improvement in a much shorter time. Companies can use a subset of development projects as pilots to implement and refine the new methodologies.

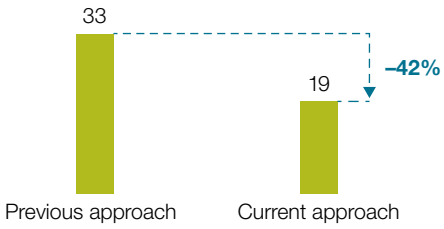
Exhibit

Excellence in embedded systems is a key performance lever.

Development time

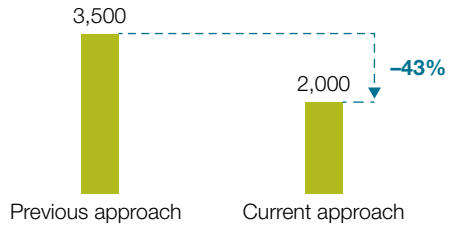
Reduced cycle time by parallelization of feature development and stabilization

Mobile-phone-platform development, months



Reduced overall project budget by improved hardware/software synchronization

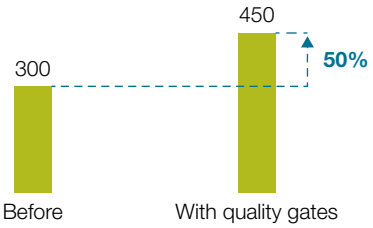
Product-development effort, person-months, comparable complexity



Defect density

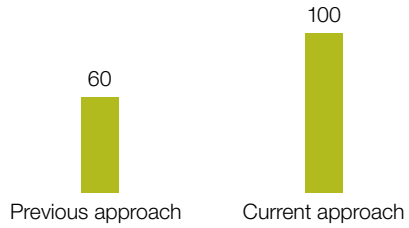
Lower defect density due to strict quality gates

Mean time between failures, hours



Significantly better predictability and quality of releases with "precommit" verification

Average release-schedule adherence, %



Companies should initially aim for a 10 to 20 percent decrease in project timelines and a significant productivity increase within the first six months. An example of a quick win would then be delivered by using a stringent requirements-definition process to ensure effective use of resources, as it drastically cuts down on rework and unnecessary development efforts.

Tighter integration of hardware and software could deliver significant benefits to many semiconductor companies. Furthermore, the measures described above have delivered significant time-to-market improvements while maintaining a high level of quality in real-world situations.

Harald Bauer (Harald_H_Bauer@McKinsey.com) is a director in McKinsey's Frankfurt office, and **Ondrej Burkacky** (Ondrej_Burkacky@McKinsey.com) is an associate principal in the Munich office.