# Getting Software into the Hardware Design Loop

By Jakob Engblom, Virtutech

It is generally accepted that software is the key to the success of current electronic products, that software development is more than half the development cost for SoC (at the 65nm technology node), and that most of the development costs for system OEMs are in software rather than hardware. This has propelled software development to front of the problems facing the chip design industry. To answer this challenge, the traditionally hardware-oriented chip design industry is striving to facilitate OEM software development. This means developing software stacks ready-made to the OEMs, and to provide ways to develop software in parallel to hardware development by using virtual platforms. However, it is possible to go beyond this and actually have end-customer software development guide hardware design, turning chip design into a truly software-driven and application-driven business.

Currently, the dominant model of chip development includes a feedback loop where market input is used to plan the next generation of a family or a platform (and all successful chip designs will be the basis for some family of platform and have a sequel, just like Hollywood movies). Together with the experience, foresight, vision, and creative imagination of the chip architects, the market feedback drives the feature set and requirements for each successive generation of chips. This has worked fairly well in the past, but as target applications start changing faster and more value is being put into innovative software, the feedback loop needs to become tighter.
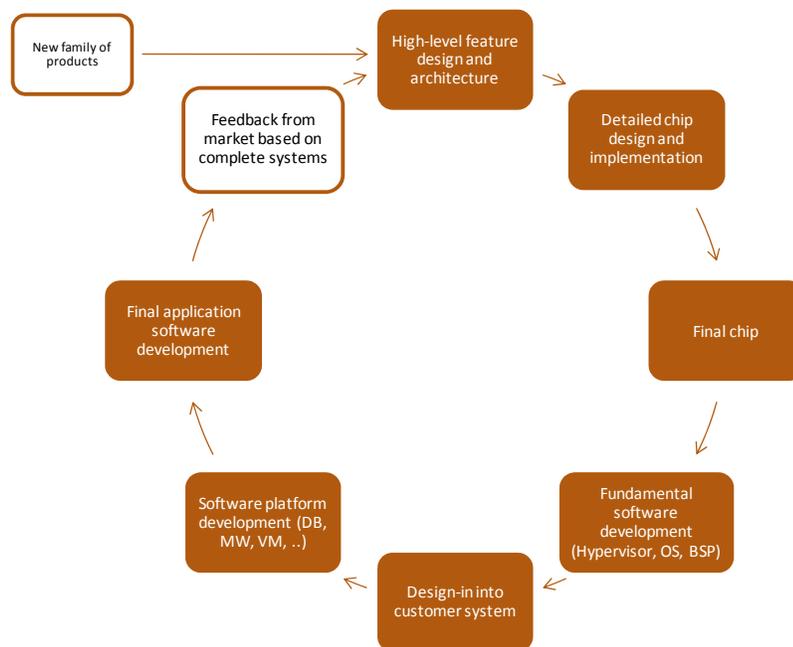


The above picture shows the long development loop that leads up to feedback from system OEMs. The chip has to be designed, manufactured, and basic software such as operating systems put on it. At this point, the sales people can probably make some customer commit to the new chip, and system

development inside an OEM (or a chain of companies) ensues where middleware, databases, network stacks, Java virtual machines, and other software components are added to form a complete platform. This system platform is then used to develop an actual application or product, and finally a complete system is ready to be tested. At this point, shortcomings or overkill in features and performance are clear, and feedback from the complete system development process is used in the product planning for the next chip.

Virtual platforms is the latest  industry solution for getting software developed in parallel to the hardware, and for validating the performance of a chip when running a particular software load. However, that usually only involves core algorithms, benchmarks, and small bodies of code from the data-plane side. The virtual platforms are mostly created by and confined to the hardware design groups, and the software design groups and end customers are very rarely involved.

To get the software designers and system OEMs really involved, virtual platforms have to reach outside the hardware design teams. This means making virtual platforms available to internal software development groups and external customers as early as possible in the design cycle. To be useful, these early virtual platforms have to be able run "realistic" software loads, in particular operating systems, so that application developers can use them to run and test application code.

As illustrated below, the result of putting a virtual platform in the hands of system OEMs and software partners is a faster and better feedback loop. It brings the "test early with end users, iterate often" principle that prevalent in modern software development thinking to hardware development. It increases our chances of developing the right thing for the market, at the right time.
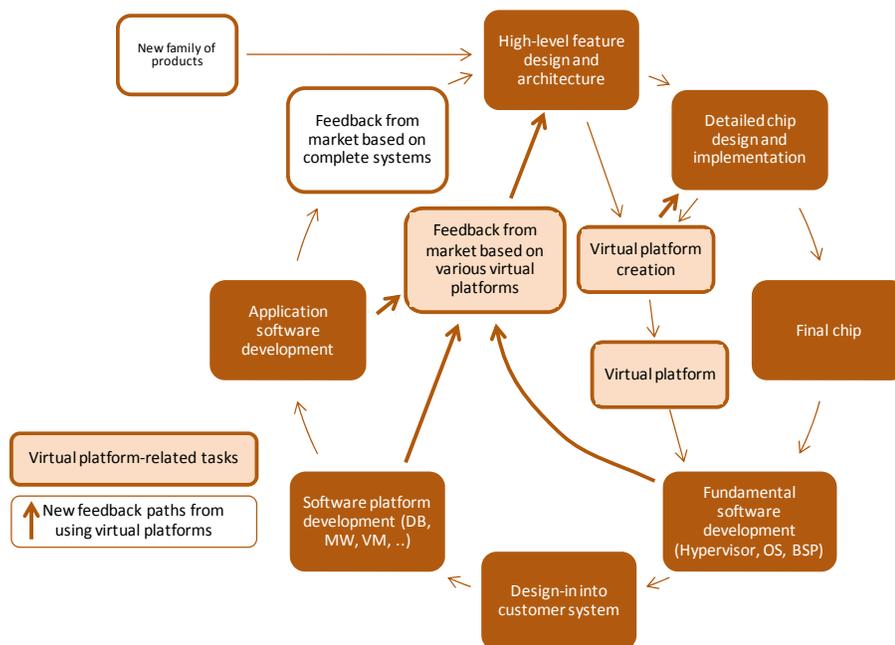


If all hardware models and supporting software had to be developed from scratch for each new chip, getting a virtual platform and supporting software out to application developers and system OEMs
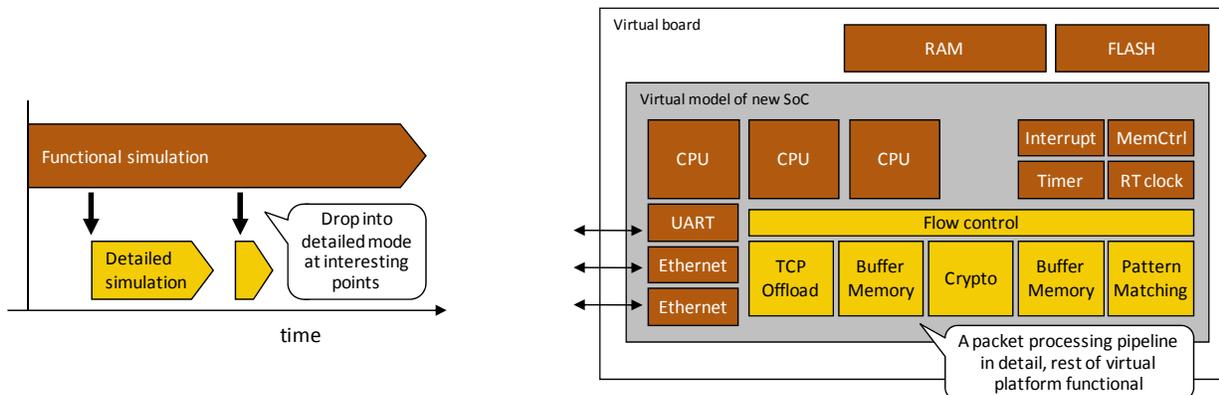
would be very time-consuming. However, in practice, almost all new chip designs reuse many elements from previous designs. This allows virtual platforms to be built quicker by reusing existing device models, and makes hardware drivers and firmware code reusable by simply adjusting the memory map in an existing board support package (BSP). Thanks to the layered nature of modern software, once an operating system runs on a particular chip, most software targeting a similar processor and operating system will also run.

Note that early virtual platform drops do not have to be complete or definite -- the very point is that they represent a sketch that can be adjusted. As soon as the basic components required to support an operating system boot are in place, you can start seeding the development of fundamental software. Today, that means a processor core (or a few), an interrupt controller, a serial port, an Ethernet connection, and memory. Such a simple platform will allow an OS kernel to be ported, paving the way for adding more BSP details and running software. This use of virtual platforms makes it possible to get feedback on the programmability and general usability of a chip very early, in time to actually adjust the design and feature set before committing the design. Areas of risk should be modeled first, such as novel virtualization support or new complex hardware accelerators.

The virtual platform approach provides feedback in two stages. First, you should make sure to have a modeling group separate from the hardware design group. Having a second set of eyes read the design documents and create an executable model of the design is a great way to test and improve the design documentation and the design itself. Second, writing software drivers for new or updated devices and integrating them into an operating system tests if the design is easy to use and works in the context of an operating system and complex software stack.

The individual device models do not need to be particularly detailed when you start seeding partners and OEMs: what matters is *what* the hardware functionality is and how you program it, not the detailed *how* of the implementation and timing. Functional models can be created much faster than detailed models and also run orders of magnitude faster. Very high execution speed is the key to enabling the software feedback loop, as seemingly simple tasks like booting an OS or running network traffic through a firewall application easily involve tens of billions of target instructions that have to be executed. Another important simplification to get device models out early is that only the most important operational modes have to be in place initially, reducing the scope of the model.
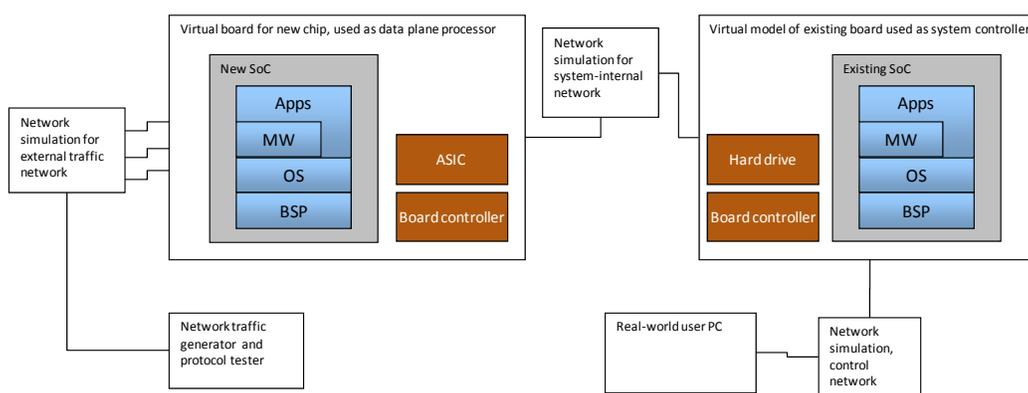
Once the software runs, it is possible to start investigating the detailed performance of a chip design, at the workload points where it is relevant. The fast functional model is used to run the software load and get the system into a state that is of interest, and then change a part (or the whole) system to use detailed models (once they are made available). The picture below shows the concepts of changing over at various points in time, and mixing models at different levels of detail within a single simulation. This is the only reasonable way to gather performance details from large software loads, as running billions of instructions in detailed mode would simply take years.

**Virtual board**

RAM | FLASH

**Virtual model of new SoC**

CPU | CPU | CPU

Interrupt | MemCtrl
Timer | RT clock

UART | Flow control

Ethernet | TCP Offload | Buffer Memory | Crypto | Buffer Memory | Pattern Matching

Ethernet

A packet processing pipeline in detail, rest of virtual platform functional

Functional simulation

Detailed simulation

Drop into detailed mode at interesting points

time

If we look at this from an organizational perspective, it does require marketing and sales to get involved with a new chip design quite early in the process. They have to contact customers and software partners, secure their support and commitment to trying the virtual platform.

Chip customers are often happy to be get involved. For a system-level OEM to be able to influence the design of the components that will go into their next-generation systems is an opportunity to gain competitive advantage and reduce their development risk. A key observation here is that by using a virtual platform in-house, they do not need to be concerned about software IP issues: all they need to provide to the chip manufacturer are the statistics gathered during the run, not their actual code. For OEMs, a virtual platform also opens up the ability to design-in a new chip before hardware availability, drastically reducing time-to-market for new products.

In order to really test their software, OEMs will usually need to extend the virtual platform with models of custom chips, specific memories, network configurations, and other virtual systems. For example, you can have a setup like the below where the generic virtual board for a new SoC has been customized to create mockups of future boards based on the SoC, and connected into a system environment built for existing products.

**Virtual board for new chip, used as data plane processor**

New SoC

Apps
MW
OS
BSP

ASIC

Board controller

Network simulation for external traffic network

Network simulation for system-internal network

**Virtual model of existing board used as system controller**

Existing SoC

Apps
MW
OS
BSP

Hard drive

Board controller

Network traffic generator and protocol tester

Real-world user PC

Network simulation, control network

It also should be noted that very early virtual platforms have other benefits than producing a better hardware design, which are equally important to help a semiconductor vendor capitalize on their new hardware design. Apart from the product management team who has obvious benefit from product design feedback, other parts of marketing can find a virtual platform very useful. It makes it possible to

get a software ecosystem in place before the launch of the chip, and to run (preliminary) benchmarks and produce marketing materials. Training materials can be developed and tested before the chip becomes available.

For sales teams, a virtual platform is a great sales tool when used correctly. It can be used to prove to customers that design decisions that look "odd" are actually the right thing to do, and that the product strategy embodied in a design is actually sensible. It makes it possible to go before key accounts and actually show them a new product in action, rather than just a set of slides extolling the greatness of the promised SoC (especially nice when the competition only has slides). The use of a virtual platform also makes it possible to perform the design-in work while the hardware is being finalized, shortening the time to volume shipments and the real sales commission. A virtual platform can also overcome the initial ramp-up phase of a new chip where development systems tend to be in tight supply, limiting the ability to engage with customers.

To sum up, chip designers need to get feedback from the software people must faster than they get it today. The way to get there is to provide virtual platforms to not just internal development teams, but also marketing, sales, and external customers. This will lead to better-designed and architected chips and chip families, and more satisfied end customers.

*Jakob Engblom is Technical Marketing Manager at Virtutech in Stockholm, Sweden. He holds an MSc in Computer Science and a PhD in Computer Systems from Uppsala University. He has been with Virtutech since 2002, and has worked with embedded and real-time systems software and programming for the past decade. See http://www.virtutech.com, and http://jakob.engbloms.se.*