White Paper

Hardware/Software Co-Verification Using FPGA Platforms

Introduction

The problem of hardware and software co-design is as old as systems design and the integration of systems composed of multiple elements. Systems built using electrical and electronic subsystems, mechanical subsystems, software, and embedded software and firmware have always been difficult to design. The problem was less complex when technology for mechanical, electronic, electrical, and software did not allow paper designs to be realized easily in a tangible system, but things have changed with the increasing complexity of systems.

Not doing staged integration and co-development of hardware plus software is extremely costly because of the added complexity. Parallel development efforts must be run for the subsystems and the cost of errors along the way minimized. This has become important due to:

- *Time-to-market pressures,* such as launching the product before competitors do, before the next shareholders call, or because customers are asking for it.
- *High cost of custom semiconductors:* ASIC development is extremely expensive. For lower power, the latest geometry node and process are necessary, while high performance requirements dictate the building of custom solutions for area and power savings and for maximum performance.
- *Cost of re-design and sustaining:* Product design cycles are typically shorter than re-design and test phases, so sustaining becomes a costly exercise for poorly designed products.

These factors have spawned many technologies and tools for process management at the project and design level, as well as integrated design flows at the element or component level such as ASIC design. The flows and technology have matured much faster for software development and ASIC development, as there are numerous instances, corporations, and groups involved. Complex system design requires staying power to manage an entire product design cycle involving hundreds if not thousands of constituent elements.

In contrast, FPGA-based prototyping platforms enable both a wider adoption of development methodologies and risk and cost reduction in integrated product development.

Quantifying the Problem

Figure 1 shows a development flow for a hypothetical product that has a mix of hardware and software with custom elements. The development flow goes through eight stages:

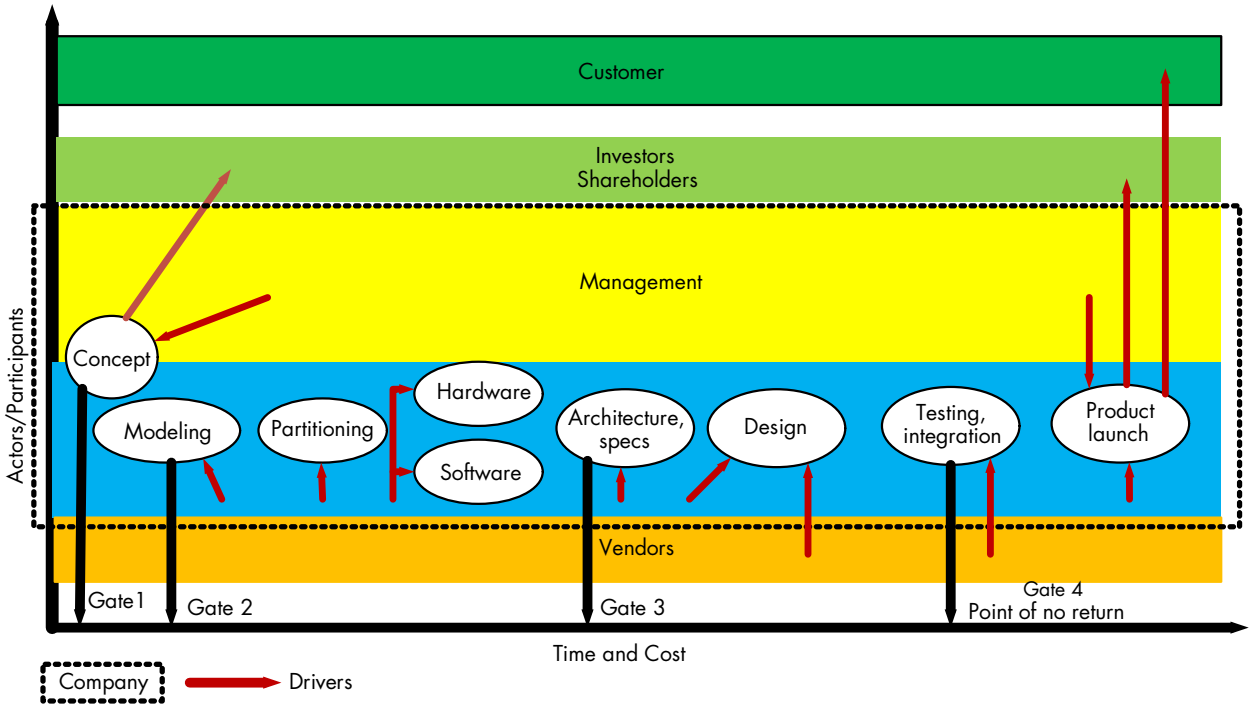
1. The concept for a product is developed jointly by management and engineering, and is communicated to the shareholders and investors as a preview of future business plans or a roadmap. This concept stage serves as Gate 1 for a go/no-go decision.

2. Engineering and marketing validate assumptions in the concept by modeling, which leads to Gate 2 for a go/no-go decision. If the feasibility of the concept is found to be impractical because of the premise or assumptions, a refinement of the concept (return to the Stage 1) or no-go decision are likely. The output of this stage is an equivalent to a high-level requirements document (HRD) coupled with a market requirements document (MRD). These serve as the feature and function descriptions for the engineering and execution team.

3. After a go-ahead decision at Gate 2, the systems engineers and architects begin to partition the concept and model into hardware and software components. They also set the top-level performance constraints. The typical output of this stage is the systems design document (SDD).

4. Part of the process at this stage is the design and architecture exploration.

Figure 1. Actors and Drivers in an Integrated Product Development Flow



5. Gate 3 comes after the team leads, individual contributors, and hardware and software systems engineers generate a detailed architecture of what is to be implemented. The output of this stage can be thought of as an equivalent to a high-level design document (HLD). In addition, trade-offs are made about what can be done within budget, with state-of-the-art technology, where all the elements come from, and within project plan and timeline restrictions.

6. After a go/no-go decision at Gate 3, the active design process begins. The internal teams and vendors are engaged for a variety of services (providing software services, parts, testing, etc). At this stage, the company still can decide to cancel product development without going through a detailed engineering and execution phase.

7. There must be a final or factory test and integration before product launch. Typically, once a project has reached this stage, enough dollars have been sunk into it that it is beyond the point of no return. In rare cases, projects are cancelled at this late stage; despite the money that has been spent, other factors influence the decision.

8. The product launch is where the engineering team, management, and all aspects of the value chain are validated by customer and market acceptance.

Note that even though cost is shown on a linear scale, it goes up exponentially as number of people involved and the budget outlay for operations and capital increase as time passes. The tangible and intangible costs of failures, errors, and delays scale exponentially with time.

Potential Solutions

The consequences and the cost of discovering errors and performing modifications associated with feature functionality required in product go up as time progresses. These include:

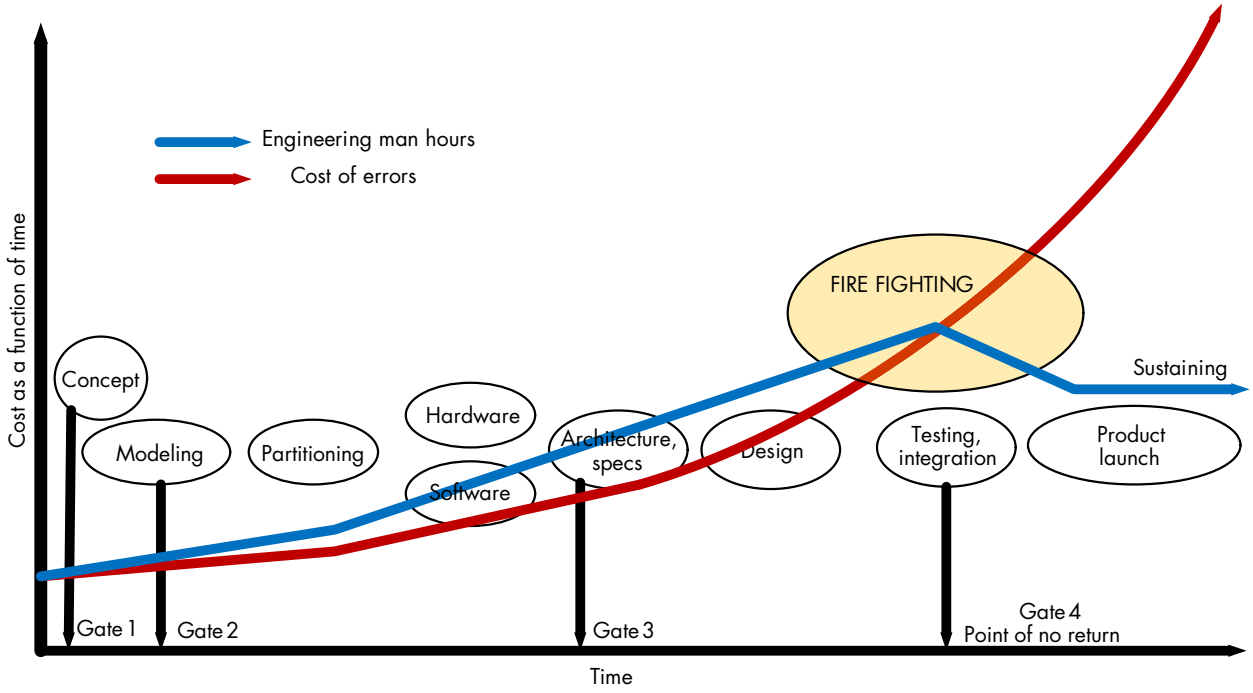
- *High launch and sustaining cost:* The high cost of discovering functional or feature-related errors in silicon or system operation leads to a much higher sustaining cost for the product (recalls, re-design, support, maintenance, loss of market share, etc.).

- *Increased planning complexity:* Planning integrated product development can be difficult where hardware, software (applications), and embedded software development are run in parallel tracks. Staged integration and co-verification with robust feedback does not take place. The hardware takes longer to produce and modify, as the whole process is serialized no matter that it is an ASIC, system, or subsystem. The software and firmware (embedded software) teams wait for the hardware to test the drivers, feature sets, hardware ion layer (HAL) and other features and functions. This reduces the probability of first-pass success, increases risk, wastes development dollars, and delays the schedule.

FPGAs, especially Altera® Stratix® III FPGAs, and design software such as Quartus® software, provide a clear risk reduction solution. IRIS Technologies incorporates multiple Stratix III EP3SL340 devices in its S3 card to provide a standards complaint (form factor and I/O), scalable prototyping platform for ASIC and systems prototyping, and simulation and emulation. The S3 cards and IRIS systems use standard design flows and tools from Altera and its partners. The "building block" method of putting together systems and platforms enables rugged, robust, scalable solutions. This paper examines several use models by application, the key features of the S3 card, and the advantages they provide for each application for ASIC and systems prototyping, simulation, and emulation.

Figure 2 shows cost as a function of time overlaid graphically with the product flow.

Figure 2. Cost of Errors and Modifications as a Function of Time



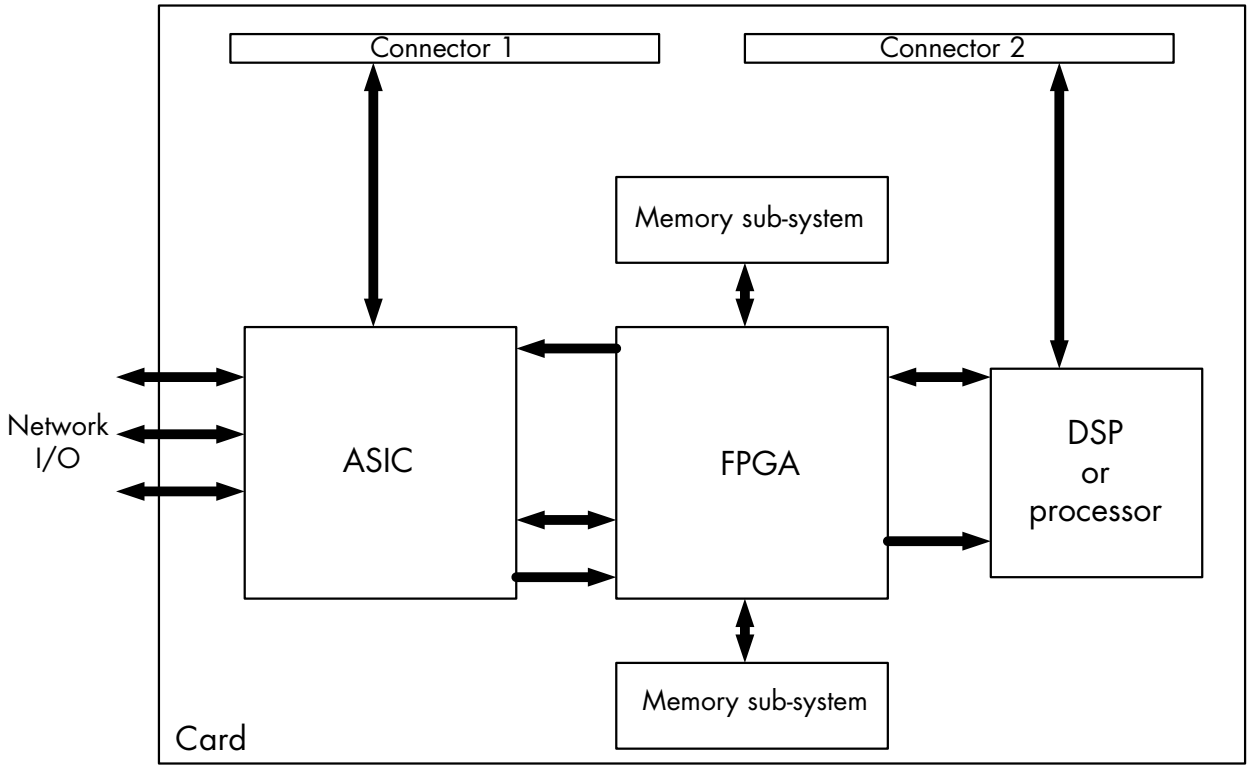
Every project manager, engineering manager, and executive tries their best to avoid the "fire-fighting" zone. The solution to avoid this is multi-faceted, but a key enabler of any risk reduction strategy is the availability of a reconfigurable prototyping platform that can be used for hardware and software co-verification. Good rules to follow in an integrated product development flow are:

- Form strong integrated product teams with systems engineering skills that know the application well.
- Test each software and hardware module in-system as part of the flow to minimize risk. Provide hardware-in-loop (HIL) testing, a good QA- and module-level test and verification plan.
- Perform incremental integration, verification, and testing of hardware and software modules using emulation and prototyping platforms.

An Example Target Design

Figure 3 shows an example design-possibly a line card, a set-top box, or a generic motherboard-that involves hardware and software components. It has software at the application layer, embedded software (firmware), perhaps a modified operating system kernel with a protocol stack, and hardware that consists of an ASIC, an FPGA coprocessor and an off-the-shelf digital signal processor or standard processor. The operating system and the protocol stack run on the ASIC, the FPGA serves as a memory interconnect and bridge, and the processor runs embedded software (firmware).

Figure 3. Example Design (1)



Note:
(1) This design is not a representation of the limits of the FPGAs and the platforms discussed

The block diagram in Figure 3 is a result of the partitioning stage of the engineering design flow in Figure 4. The block diagram is part of a high-level block diagram created for the MRD and HRD. Stage 1, target emulator event-driven modeling, Stage 2, proto-platform ASIC and system, and Stage 3, integration platform software test and firmware test are critical for risk reduction and staged integration of all the modules in the product development flow. They also help in the smooth integration and testing of the first article and moving it to the production and release stages.

To run parallel hardware, software, and firmware efforts, each requires a verification and validation test bed. Test benches and test environments help in functional testing and generation of golden test vectors for actual hardware testing. The integration of software and firmware cannot wait for the ASIC to be ready and verified, or for the PCB and components to be designed and tested. If any errors or modifications are required, then the process is in the fire-fighting zone.

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/948136031110006113