# System Emulation and Digital Twins in Aerospace Applications

Frank Schirrmeister
Systems Verification Group
Cadence Design Systems, Inc.
San Jose, CA, USA

*Abstract* — **Over the past decades the complexity of systems to be designed has grown at breathtaking pace. What may look to an individual design team like a system, almost certainly turns out to be a component in a yet bigger system.**

**Individual, licensable blocks of semiconductor IP like processors, peripherals and buses are integrated into licensable subsystems. In turn, these become components in Systems on Chips (SoCs), with their miniaturization following Moore's Law and productivity improvements keeping design cost affordable through tools and re-use enabled by the Electronic Design Automation (EDA) industry. SoC's in turn become components on Printed Circuit Boards (PCBs) in products, that get integrated into systems like the F-35 Fighter jet. While the F-35 is arguably the most sophisticated machine on the planet, integrating 200,000 parts from 1,600 suppliers using 3,500 integrated circuits (ICs) and 200 unique chips with more than 20 million lines of software code, the F-35 becomes "just a component" of the yet bigger system of the air traffic and communications network.**

**System emulation bears the potential of improving pre-production quality with first time success of designs, allowing to collapse the serial development pipeline, speed up of verification run-times and allowing easier and earlier distribution of platforms for software development, not to mention that increased safety during testing as a crash in a simulator is much better than putting real lives at risk. System Emulation also has significant overlap with what the industry calls Digital Twin, virtual representations of the system to which the same stimulus can be applied.**

**This paper analyzes the challenges for system emulation for electronic devices and systems. Using practical user examples from the commercial SoC world, we will outline the benefits emulating systems of various scopes and levels of abstract early in the design flow, using virtual and physical emulation.**

***Keywords—system emulation, digital twin, virtualization, system on chip, SoC, verification, emulation, simulation, FPGA based prototyping, functional safety, aerospace, Cadence***

## INTRODUCTION

The constant growth in the complexity of designs seems to be so fast and staggering that the item we are working on at any given point in time seems to be at or just above the complexity that can be handled.



Figure 1 shows a typical "systems of systems" challenge. Key themes like big data and artificial intelligence, decentralized, human and robotic systems, transportation and global tech and infrastructure work hand in hand and co-exist.

Fig. 1. Systems of Systems Scenarios on land and air

A SoC a very complex system in itself—becomes just a component within something bigger, for example a networking environment. The embedded processor cores that execute the software in embedded systems are already very complex; then the processor becomes a component of the compute sub-system in an application processor, which then becomes a component of the SoC that is integrating the compute sub-system with custom accelerators, sensors and actuators. The SoC, again, is just a component of the actual electronic system integrated on a board, which is a component within a network of distributed control units. Bottom line, in a car or plane, each electronic control unit (ECU) is made of components and is also a component within the network that is the car or plane, or even its environment.

The impact of each hardware and software component, starting from IP blocks, sub-systems to SoC that are connected in PCBs and system, with its impact on overall system functionality and performance need to be carefully considered, see also [1]. Engineers also must consider full-system factors, such as thermal effects and interaction with mechanical actuators, as well as the interaction between systems.

The development challenges for Systems on Chips (SoCs) have long been driven by two main trends, increased miniaturization and ever-growing system complexity. Moore's

Law has allowed designers to integrate more and more, ever smaller components per chip, targeting ever shrinking technology nodes now approaching 7nm technologies. On the other hand, more and more building blocks of a design that were often previously connected as discrete components on a printed circuit board (PCB) are now getting integrated into a single chip, having created a striving market of reusable silicon intellectual property as often 70% or more of an SoC are re-used.

SoC-design for the aerospace application domain adds several additional challenges to the already complex development, verification and validation of systems on chips, namely compliance with functional safety standards and system environment complexity. Like in other application domains including automotive for autonomous driving and automated driver assist (ISO 26262), industrial for the needs around factory automation (IEC 61511), medical for robotic surgery and implants into the human body (IEC 62304), functional safety has become a critical issue in control systems of trains (EN 5012x) and flight systems (DO-178, DO-254) and needs to be considered during development. In addition, due to the sheer system complexity in aerospace applications, verification needs to consider the different domains involved – flight control, cabin and passenger – and has to even put them into the context of multiple airplanes and ground control systems.

## System Emulation and Digital Twins

One key trend in aerospace and defense applications is the desire to create Digital Twins and a Digital Threads of the systems during the development phase and during the system's lifetime. As outlined in [2], the definition of those terms is not very concise and in the industry one can find many at times confusing definitions for both, and the "concepts mean different things to different stakeholders within the ecosystem. OEMs and suppliers have one point of view while owner operators, commercial players and military have their own. Service providers have yet another definition.". The suggested definitions in [2] are as follows:

- A Digital Twin is the digital representation of the "current state" of a manufactured product or system at any given point in time.
- A Digital Thread is the digital record of all "states" of a manufactured product or system over time from conception to disposal.

For the purpose of comparison with "System Emulation" we are considering a Digital Twin to be defined as follows:

- A Digital Twin is the digital representation of a product or system under development representing a functionally correct, predictable and reproducible representation of the product or system at the appropriate level of fidelity to perform verification, performance analysis and system validation tasks.

Figure 2 shows a typical project flow during a system on chip hardware development, from idea to chip availability. The timelines are averaged from several projects that have been analyzed by IBS and Cadence. A hardware stack from reused blocks ("IP") that are integrated in sub-systems, systems on chip and eventually into a system in a PCB board is being developed and integrated with software as indicated in Figure 3.

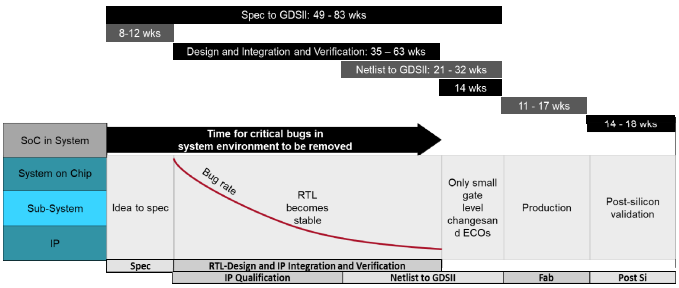


Fig. 2. System on Chip development flow

After several weeks of spec development, the bulk of the development time is spent on verification of the RTL in the context of re-used IP blocks, and on the implementation from RTL into GDSII representation that can be manufactured, which adds a constant time of 11 to 17 weeks. Once silicon is available, it needs to be brought up in post silicon validation. Functionally, about 60% throughout the project, all defects have to be removed as from there functional changes are no longer possible.

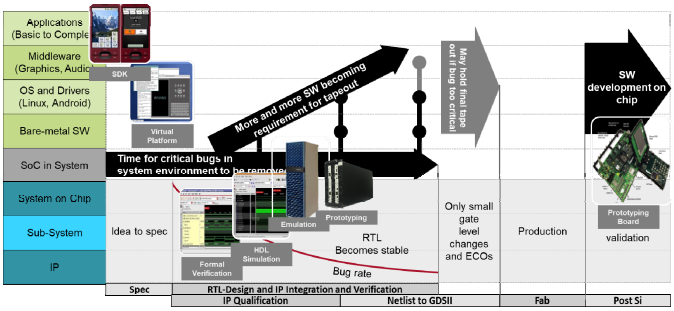


Fig. 3. System on Chip development flow

Figure 3 adds to the hardware development flow described earlier the software development stack from bare metal software to operating systems, drivers, middleware and applications. It also illustrates different representations of the hardware under development during a project flow that can be used for software development.

Each function in the system (both inside and outside the chip), can be described at different levels of fidelity using different languages and different execution engines. All of them represent different "Digital Twins" in the context of chip development, offering different levels of insight and fidelity to reproduce issues that are found in the actual system later on and to allow software development and integration as early as possible.

Software Development Kits (SDKs) typically do not run the actual software binary but require re-compilation of the software. The main target users are application software developers who do not need to look into hardware details. SDKs offer the best speed but lack accuracy. The software

executing on the processors, as in the SoC examples given earlier, runs natively on the host first or executes on abstraction layers like Java. Complex computation, as used in graphics and video engines, is abstracted using high-level APIs that map those functions to the capabilities of the development workstation. Typical examples from the consumer world are Android Development Kits, the iPhone SDK and in the aero/defense space the WindRiver VxWorks SDKs.

Software-based virtual prototypes run the actual software binary without re-compilation at speeds close to real time – potentially hundreds of megahertz. Target users are software developers, both apps developers and hardware-aware software developers. Depending on the need of the developer, some timing of the hardware may be more accurately represented.

While these virtual prototypes are often used at the SoC level, [3] paper presents the early deployment of a fully virtual platform called SIMUGEN to perform the tests of certified airborne software as an alternative to the current approach based on the use of dedicated hardware platforms. Expanding the simulated scope even further, [4] illustrates a model based approach for complex avionics systems that can be applied to the level of complexity of cabin management systems.

RTL simulation executes the same hardware representation that is later fed into logic synthesis and implementation. It may only execute in the single-hertz range, but it is hardware accurate as the RTL will become the golden model for implementation, allowing detailed debug.

Verification acceleration executes a mix of RTL simulation and hardware-assisted verification in an Emulator, with the test bench residing on the host and the design under test (DUT) executing in hardware. As indicated by the name, the primary use case is acceleration of simulation. This combination allows engineers to utilize the advanced verification capabilities of language-based test benches with a faster DUT that is mapped into the hardware accelerator. Typical speed-ups over RTL simulation can exceed 100x and are naturally limited by the relationship of how much time is spent in the test-bench vs. the DUT. An example of emulation for simulation acceleration can be found in [4].

Classic In-Circuit Emulation executes the design using specialized hardware – verification computing platforms – into which the RTL is mapped automatically and for which the hardware debug is as capable as in RTL simulation. Interfaces to the outside world, such as Ethernet and CAN, can be made using rate adapters.

FPGA-based prototyping uses an array of FPGAs into which the design is mapped directly. Due to the need to partition the design, re-map it to a different implementation technology, and re-verify that the result is still exactly what the incoming RTL represented, without proper automation the re-targeting and bring-up of an FPGA-based prototype can be cumbersome and hardware debug is a difficult process. In exchange, speeds will go into the tens of megahertz range, making software development a realistic use case.

All the techniques above are applied prior to silicon availability and are typically referred to as "verification". They are mirrored in the post silicon development phase with techniques for lab and production testing.

Once a chip is available as Silicon Development Kit, software development can commence on the actual production hardware and lab testing starts. Traditionally kept separate, there is value in linking simulation to silicon to combine post-silicon testing techniques with pre-silicon verification, allowing a shift-left of testing into the pre-silicon phase and efficient re-use of pre-silicon verification aspects post silicon availability.

There is literally no "one fits all" representation that meets all requirements that development teams may have during the development cycle, so a Digital Twin of a new design will almost certainly combine different engines to allow verification, analysis of performance and power characteristics, software development and in-system validation.

Some of the key characteristics are shown in Figure 4.

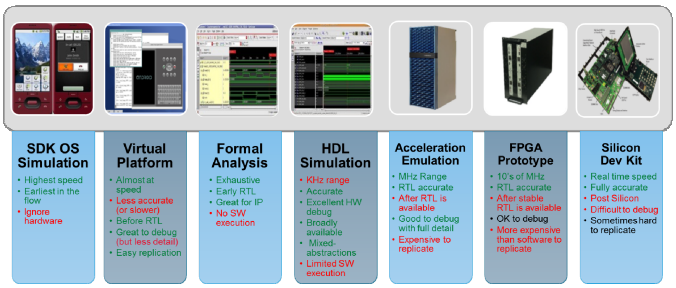


Fig. 4. Different development engines for hardware software designs

In classic development flows these different representations come and go and once a project proceeds are not maintained to represent and include changes made in later project cycles. In the era of "Digital Twins" at least some of them are now maintained and updated to the latest changes to allow reproduction of functionality experienced in the actual system. The effort for this type of maintenance has to be carefully weighed against the return of investment they can provide to reproduce defects fast when they occur, or to develop and apply new scenarios to a design before they are tried in reality.

## I. System Emulation Considerations

Emulation is often used as "for usage as substitute", in contrast to simulation that is usually used "for analysis and study" (see [5]). Common usages are prototyping in FPGAs, simulation of scenarios, hardware in the loop of simulation execution, emulation of digital circuits and SoCs to achieve acceleration over simulation and virtual prototyping and emulation at higher levels of abstraction. Independent of the scope of the system to be emulated, proper choice of fidelity for each component becomes a critical issue (see [1]) and often leads to the need for mixed fidelity execution.

The description of a fully "executable specification" of the system and all its components is desirable and some progress has been made as illustrated in [4]. However, its application at the mission level has limitations due to the abstraction in model-based descriptions.

Similarly, at lower levels of abstraction, while an array of hardware-based emulators is an intriguing concept to execute a full car or airplane, it would have many challenges. For instance, just having all the components available for emulation at the right time is a challenge. Some components will be available in silicon already, some will be still conceptual and only representable as C-models, others may be available as RTL for emulation.

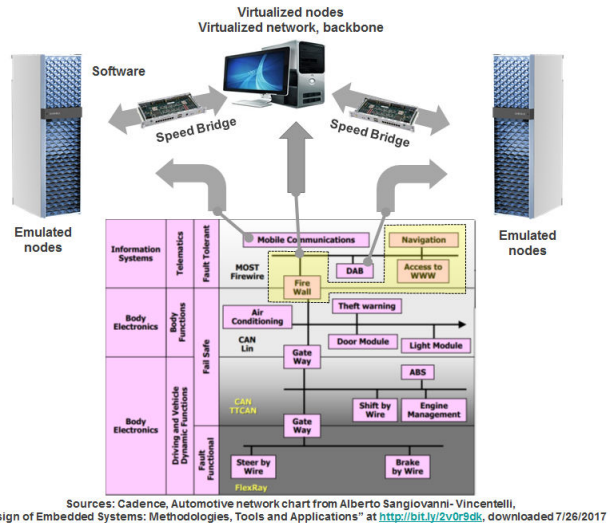


Fig. 6. Setup for "System of Systems" execution

The alternative is proper divide and conquer—abstracting away some aspects that do not directly impact the component under development. The illustration in Figure 6 shows an example setup combining several nodes of a system at different abstraction levels, combining real and virtual nodes. This is a natural extension of what the industry calls "hybrid emulation," in which the processor sub-system of an SoC is abstracted into a virtual platform at the transaction level.

In the example above, some nodes of the system are emulated, connected via rate adapters with some of the nodes being virtualized running as software on a host. It's a true mixed abstraction set-up, and allows verification and validation of the components that are emulated.

Both "digital twins" as described above and in [4] can exists in parallel and can have different application spaces based on their characteristics and the fidelity they provide.

## II. Comparison to Consumer Electronics

During the 2018 ERI workshop in July in San Francisco [5], several questions specific to emulating systems of systems were discussed. Emulation was defined quite broadly as one of the following representations:

- Virtual Models, like QEMU
- Digital Circuit Emulation using hardware based emulation like Cadence Palladium
- FPGA Based Prototyping
- Rate Adapting Hardware in the Loop
- Scenario Simulation

The fundamental benefits of those representations include improvements in pre-production validation quality leading to first time success of SoC and system design and the ability to collapse the serial development – the industry refers to it as "Shift Left". Emulation also has the potential to speed up verification run times leading to faster time to market, can in its software-based incarnations be much easier distributed to users. Finally, it is much better to crash in a simulator/emulator than in the actual aircraft.

In the discussion four basic questions were asked

- Is real-time emulation possible?
- Is "in-the-field" emulation possible?
- Are standard interfaces required?
- Is virtual qualification possible?

The possibility of real-time emulation really depends on the use case. In some areas like medical applications the speed of the end device may be slower than emulation and as such real time usage is possible. In most cases, emulation techniques run in the MHz range for hardware-based emulation, in the 10s of MHz range for FPGA based prototyping and in the 100s of MHz range for virtual prototyping. As such they are in most cases slower than the actual end device, but with custom silicon it may be possible to develop application specific emulators that execugte close to or at real time.

Whether emulation in the field is possible depends on the scope of the system to be emulated.

Airplanes with physical emulators that have specific power and weight requirements onboard to test how different control algorithms would work in real time generally are unlikely. In contrast, the alternative method of virtualization the airplane and using real traffic data is often not fast or accurate enough to reflect the appropriate real-world effects.

It is quite common in the automotive world to drive actual prototypes in previous generation cars to understand how they behave under real conditions, but the industry is also applying "Virtual Hardware in the Loop" approaches that allow execution without having the actual hardware available. This is also happening in the communications area - for instance, Sirius XM will use prototypes for field tests [6]. While field tests will always be needed as part of the overall set of tests that are executed, it is possible to collect trace data from field tests and apply them in the lab to virtual protypes, emulation or FPGA based prototyping.

Balancing the fidelity of the prototype with speed and cost becomes a critical issue to look at. Digital twinning also plays a role as users can collect data in the real world and apply it to a digital twin of an airplane or car.

With respect to standard interfaces, standardized software stacks for self-driving cars may be a critical element for proliferation of technologies. In the scope of SoCs, interface standards like CAN, Ethernet AVB, and others are already

largely standardized. Software standards like AUTOSAR help re-shape the design chain and dependencies between suppliers and OEMs.

Finally, given the criticality of obsolescence, virtual qualification of electronic components is a promising area. System emulation can be used to re-apply data to a part that had to be re-developed because of obsolescence. There are interesting future aspects to be considered that are extending beyond pure electronics – for instance do the parts work in all angles and all acceleration specs that were defined? Users can re-apply the same tests that were used for qualification years ago and perhaps qualify the new part easier

## III. Results and Outlook

System emulation has a fascinating future ahead in aerospace design as we are facing true "systems of systems." Digital Twins are specific instances of system emulation with specific use models, allowing to apply real data to a virtual representation of the emulated system.

Examples from the automotive, consumer and communication domains can be easily transferred to aerospace applications, even allowing development using Digital Twins using emulation engines combined with classic software-based verification techniques.

## References


[1] Towards "Systems of Systems" Verification for Systems on Chips in Aerospace Applications, Frank Schirrmeister, George Zafiropoulos, GOMAC Tech 2018

[2] Establishing a Fully-Functional Digital Twin or Digital Thread in Aviation, Aerospace and Defense, David Grasso, December 8, 2017, http://bit.ly/2rlgtpq

[3] Airborne software tests on a fully virtual platform, Famantanantsoa Randimbivololona, Abderrahmane Brahmi, Philippe Le Meur

[4] Automated Validation of Minimum Risk Model-Based System Designs of Complex Avionics Systems, Nils Fischer, PH.D. Thesis, Fakultaet fuer Informatik und Automatisierung, University of Ilmenau

[5] "Hardware Emulation Workshop", DARPA ERI Summit, July 23-25 2018, San Francisco, CA, http://bit.ly/2xoL45n.

[6] "DAC 2017: A Glimpse Of How The Future Is Enabled", Frankly Speaking, http://bit.ly/2lh7oLu, downloaded February 2019.