Acceleration of product and test environment development using SystemC-TLM

[*]Florian BARRAU, Alexandre PICCINI, Alexandre NABAIS MORENO
[†]Mark BURTON, Luc MICHEL, Clement DESCHAMPS
[*]Schneider Electric - 2 Chemin des Sources - 38240 Meylan
Email: florian.barrau@schneider-electric.com
[†]GreenSocs - 46 avenue Felix Vialet - 38000 Grenoble France
Email: info@greensocs.com

***Abstract*— While SystemC/TLM-2.0 is an established technology allowing the parallel development of both hardware and software components of a platform or product, less emphasis has been placed on the ability to use the same technology to develop test environments. Test Environments have become increasingly complex, tracking the complexity of products; they have become products in their own right. This paper demonstrates the benefits of using SystemC-TLM Virtual Platforms for preparing test sequences before having real hardware. The proposed approach involves two software-based virtual platforms: for the product and the test equipment. Specific libraries and drivers have been developed for TestStand (National Instruments test sequence manager) allowing communication with a SystemC/TLM virtual platform modelling the test equipment. Early results already show the expected benefit of the individual Virtual Platforms, finding bugs early in firmware for instance, and show an overall acceleration of test development of several weeks.**

Virtual Platform; SystemC-TLM; TestBench; Verification; TestStand

I. Introduction

Product verification guarantees a device functions reliability for the operations for which it has been designed. As connected products increase in complexity, the scope of the potential design space that must be verified for a product has increased many fold. Verification teams have to consider all the product use cases, including how they are connected to other products, the complexity of which does not cease to increase. The time taken to perform meaningful verification has always been significant, as the design size increases this becomes an important issue. Therefore, it has become mandatory to anticipate the verification campaign for a product as early and in as much detail as possible. This helps the time to market.

Traditional ways of testing products often result in waiting for the product to be available before starting to implement the tests. The tests deal with the validation of the interfaces, and inject specific signals that are coming from hardware components e.g ADC,GPIO, which will possibly be interpreted by firmware.

Many existing bug localization practices rely on failure reproduction once the hardware is actually available. This is time consuming, and potentially costly. It becomes very difficult for complex designs to validate the product, especially when faced with non-deterministic behaviours such as interrupts, interaction with multiple processor cores, and electrical interference.

This paper reports on the use of Virtual Platforms, a methodology which allows software to be developed without access to the real hardware. In our case, this allows test sequences to be prepared without having the device or hardware testbench available. Introducing two virtual platforms, one for the device itself, and one for the testbench, enables our teams to work together early in the design cycle. Overall this helps reduce risk, and shorten development time.

We demonstrate the effectiveness and practicality of the method by presenting

1) A SystemC-TLM Virtual platform of the product (VP-P) that can be used to localize firmware bugs. In our existing development flows (which focus on products such as industrial circuit breakers), it can take up to 3 weeks for the delivery of a hardware development board, where firmware is developed. It's roughly the same when we move from the development board (starter kit) to the final product hardware board with real interfaces. Using a Virtual Platform we can start working immediately, without any delays.
   A typical virtual platform contains several peripherals which are modelled using SystemC-TLM, while the processors are emulated using QEMU. The virtual platform has a handshake mechanism between QEMU emulating the processors, and SystemC managing the scheduling of different peripherals such as the GPIO, ADC, GMAC, DDR, etc.
   Figure 1 shows the main peripherals modelled. Peripherals marked with a dashed line are capable of communicating with the real-world. This makes the virtual platform very powerful, as it can communicate with real-world devices. This allows products to be simulated in real world settings prior to H/W being available.

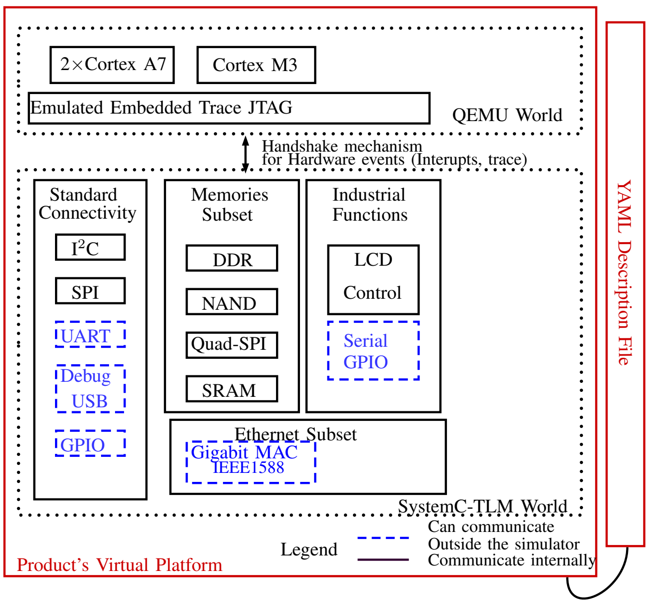


Fig. 1. Virtual Platform of the Product

   While typical application code in safety critical environments are often generated through flows, the firmware that supports those applications must be written (by hand) for the specific platform concerned. Our methodology helps to find inconsistencies between the hardware design and the firmware, and between the firmware and the application code. A typical bug in firmware causes writes to incorrect hardware registers (possibly due to a misunderstanding of the register map, or the need to initialize certain registers). This sort of behaviour can be critical, especially on real hardware, as security keys stored in registers may be inadvertently over-written, rendering the board inoperable.
   Using a VP-P not only allows these sort of bugs to be discovered, but does so in a safe manner with no danger to real hardware.

2) Another Virtual Platform, emulating the test equipment (VP-T): Building the model of the hardware testbench forces the test and product teams to agree early in the design cycle on the interfaces between the product and hardware testbench and how that can be coupled to the test manager. Typically test and product teams may not be co-located, introducing a VP-T in addition to the VP-P not only has the direct benefits of enabling early test development but also enables the teams to work efficiently together.

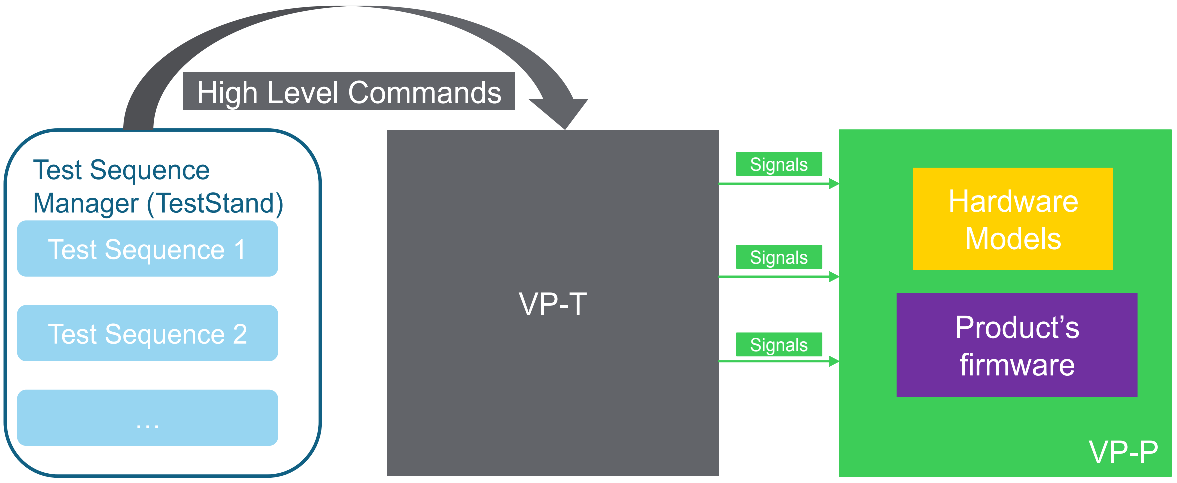


Fig. 2. Overview of the proof of concept principle using VPs

3) A proof of concept with National Instrument's test sequence manager, where verification teams start preparing their test sequences simultaneously with the teams developing the hardware and software of the product. In a classical case, the Test Sequence Manager would command a hardware Testbench that would inject signals into a product. In our case, it will command the VP-T to inject signals into the VP-P under test, as shown, in Figure 2.

A. *Motivating Example: Extend the basic V to a parallel double V-Cycle*

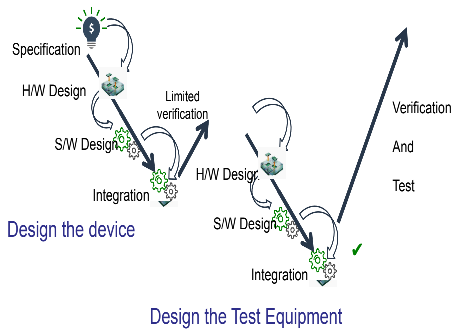
The classic V-model design flow, also known as V-Cycle, is a verification and validation model that can be used in a software development process, but is also widely used in systems engineering. [1] has shown how it can also be used for systems containing both hardware and software components, while [2] has proved its efficiency for mechanical devices.

The V-Cycle can be generalized as a multidisciplinary approach for guiding and managing development of successful systems. While System Engineering is essential for managing the complexity and re-use of different system components, the 'V-Cycle' approach has become a de-facto a standard process [3]. As products become more complex, crossing domains between digital hardware, software, analog and mixed signal, and mechanical engineering, the development flow encompasses many tools, each with their own methods and processes.

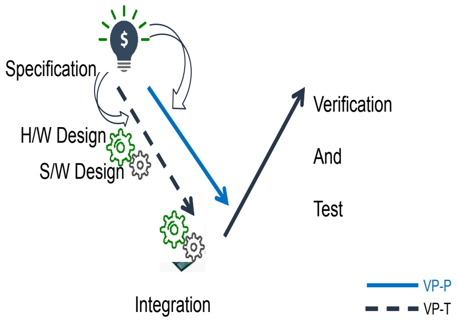
In this paper we propose to extend this approach by allowing the test equipment used to verify a product to be considered, in it's own right, as a product, subject to it's own "V-Cycle". We will show how this V-Cycle can then be parallelised using a Virtual Platforms. We will go on to demonstrate this using the relatively complex verification test sequences for an IO product [4].

In the classic design flow, the V model is applied to the product itself and the test equipment, one after the other (This is shown in Figure 3a). While hardware test-benches were simple, this may not have been an issue. Now the hardware testbench and the test sequences that run on it have become much more complex, it is now possible to distinguish two separate V-Cycles. Adding a virtual platform to a V cycle (as shown in Figure 3b) improves time to market, and communication between the teams involved, reducing the risk of a re-spin.

Introducing two Virtual Platforms, one for the product (VP-P), and the other for hardware testbench (VP-T) allow testers, hardware designers, firmware developers and architects to all work together simultaneously without waiting for the different parts. On the one hand, firmware engineers dont need to wait for hardware to start developing. On the other hand, testers dont need to wait for the product with appropriate interfaces to start developing their testbench. The 2 Vs collapsed and allows these two different processes to be accelerated together.



(a) Test Equipmement and Product design double-V



(b) Double V merged using VP-P and VP-T

Fig. 3. Illustration of a double-V model transformed into a single one with two Virtual Platforms

II. The Testbench: Proof of Concept with TestStand

The hardware testbench uses test equipment to measure and evaluate the performance of the product.

(a) Front View

(b) Rear View

Fig. 4. Hardware Testbench - Example of hardware integration

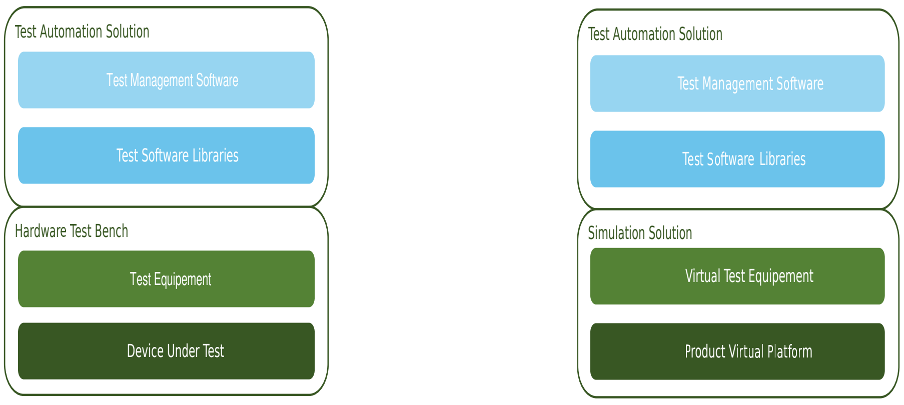
The development of a hardware testbench also follows a V cycle with the following steps: definition of the needs, technical specification, implementation, integration, verification, graduation of the required certificates of conformity and finally delivery to the end users. The duration of this development cycle varies according to the complexity of the scenarios and the expected performances of the tests. The total development time is about six months for a medium complexity test bench such as the one shown in Figure 4

In this section, we focus on automatic and semi-automatic test benches that remotely control the measurement devices and perform tests with total or partial autonomy.

A. *Test automation solution*

Software is used to describe the test scenarios, sequence them, execute them, read the measurement results, perform calculations and generate a report file. This software is called the Test Management Software (Figure 5).

The test management software relies on test software libraries which control the hardware testbench. The test sequences can be developed with standard tools or with tools specific to test benches such as NI's TestStand.The test bench is part of a test strategy, deployed for several products, at different stages of their development cycle.



(a) Classical Testbench Workflow without VP (b) Testbench Workflow with a VP of the test equipment

Fig. 5. Hardware TestBench Evolution before using VPs and after using VPs

The development of tests are not synchronized with the product development. Normally, the product specification must be mature enough to avoid any change in the hardware architecture of the test bench and to freeze the choice of the test equipment that must be integrated. Thus, test benches are often unavailable when the first prototypes of the product are released. The approach in this paper allows verification and test software development to start earlier with simulators. The verification teams can start working before having any product prototypes or hardware test benches. The embedded software of the product can be tested in its early development stages even if its design is still evolving as illustrated in Figure 5b.

This approach allows the debug and validation activities related to the development of the test sequences to start earlier. As a consequence, test scripts are more reliable by the time they are used for the real product's verification step.

B. Software abstraction and script reuse

To maximize the gains, the concept must allow the reuse of scenarios developed for the VP-T on hardware testbench used to test the real product. Automatic test scripts must be independent from the test platform that executes them. In other words, the test automation solution must remain the same whatever test equipment is used. Therefore, the test software libraries must provide the necessary software abstraction to ensure the re-usability of the test sequences. Thus, the first tests performed on the real product will probably be non-regression tests, they will validate that the hardware prototypes have the same behavior as their digital twins. The abstraction of the hardware testbench brings another advantage. There is no requirement to have advanced simulation or virtualization skills to design test scripts that will be executed on virtual targets. This facilitates the deployment of the virtual platform within the verification teams.

The test software abstraction is based on the four software layers described in figure 6

- Driver layer: abstracts the test equipment that will be remotely controlled. The underlying hardware devices could be located in the hardware or in the virtual platform. Such devices can be arbitrary waveform generators, digital IOs, measurement equipment, etc.
- Hardware abstraction layer: this layer abstracts non-controllable equipment. This category includes the compensation of threshold voltages for protection diodes, the scaling factor of a conditioning circuit for an analog signal, etc.
- Product abstraction layer: this layer deals with the specific characteristics of the product under test. For instance, the state of a digital output can be changed if the product works with reverse logic; an analog signal can be scaled in this layer to take into account the transfer function of a sensor of the product.
- Test Interface Layer: this layer gathers the inputs and parameters coming from the test management software and returns the results of the operations that have been carried out.

The software components of each layer are executed successively. Each layer must be configured so that

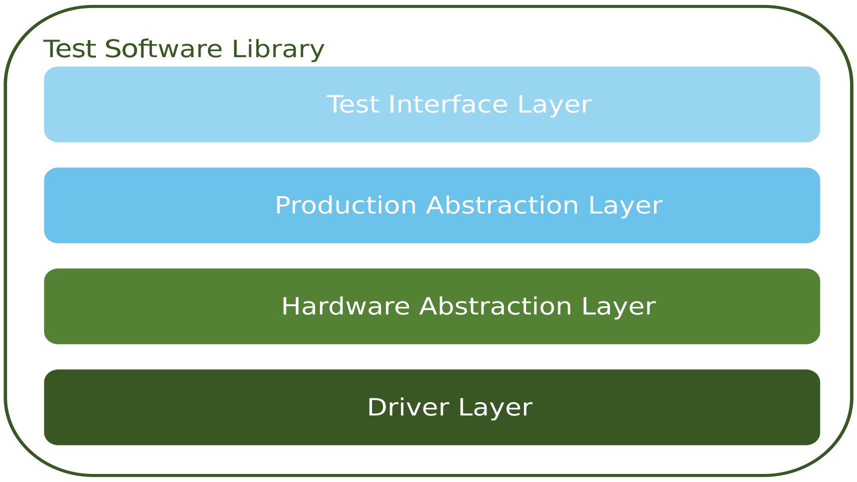


Fig. 6. Software Abstraction

the entire software solution has the expected behavior. The test management software, with its different layers remain generic. The configuration file of the automated solution is the only component that is specific to the test platform. Therefore, moving from a VP-T to a real hardware testbench can be easily done by changing this single configuration file.

III. Results

The presented process has been used with a real device. We proved the concept on a simple I/O product. VP-T showed a 2 month gain just for the preparation of the test sequences. VP-P helped firmware teams to save 3 weeks as they started working before receiving the development boards. Even when the board was available, the VP-P was still useful for debug (e.g. finding register accesses issues that are not visible on the real board). On this simple product, the time saved has been evaluated as 3 months. This is illustrated in Figure 7 and in Figure 8

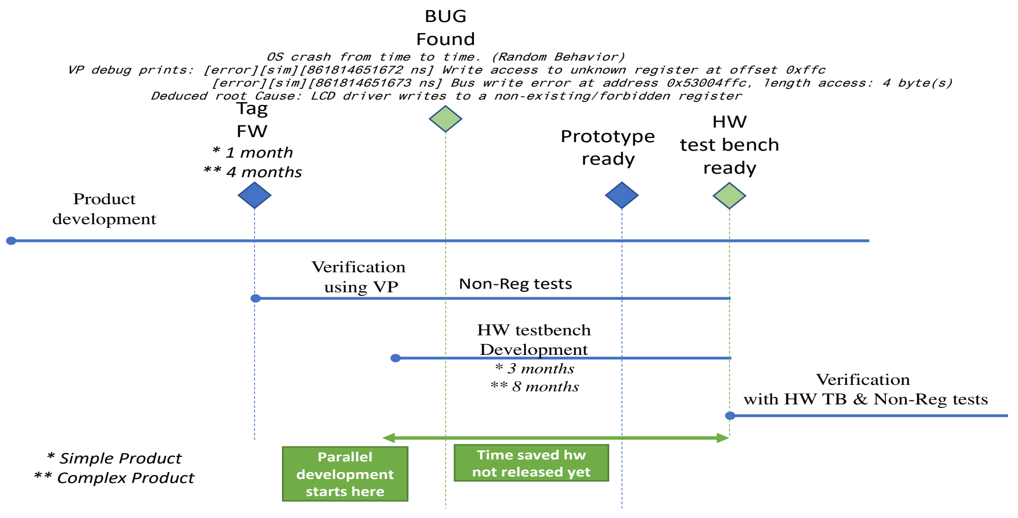

```
                 OS crash from time to time. (Random Behavior)
VP debug prints: [error][sim][861814651672 ns] Write access to unknown register at offset 0xffc
                 [error][sim][861814651673 ns] Bus write error at address 0x53004ffc, length access: 4 byte(s)
       Deduced root Cause: LCD driver writes to a non-existing/forbidden register
```

Fig. 7. Time graph when using SystemC-TLM Virtual Platforms

Figure 7 shows a graph in which a bug in the LCD driver is highlighted, that consisted of writing to a non-existing register. It is one of many other bugs found earlier thanks to the virtual platforms. This bug was critical as it led to random OS crashes from time to time.

With the VP-P we can see immediately the root cause of the issues as we are able to log access to the hardware.

As we can see in Figure 8, without the use of a VP-P, this bug in the LCD software driver might have been found, but much later. It would have been considerably harder to track and find these sort of non-deterministic bugs on a real product. Typically, bugs of this nature can cause re-spins, requiring a second version of the product's prototype. Clearly this is more expensive and time-consuming than just modifying the code and retesting it again on the VP-P.

Having a VP-T enables the development of product test sequences for product verification. The integration of the VP-T with the VP-P allows the embedded software to be executed (and bugs tracked), within the product verification environment.

In total we estimated the time saved to be 3 months for a simple product, thanks to early identification of bugs, early development of the test scripts, and early firmware development. In addition, the virtual platform is reusable for every product based on the same device. Therefore, the time taken to build and develop the virtual platform initially can be amortized over a number of product families.

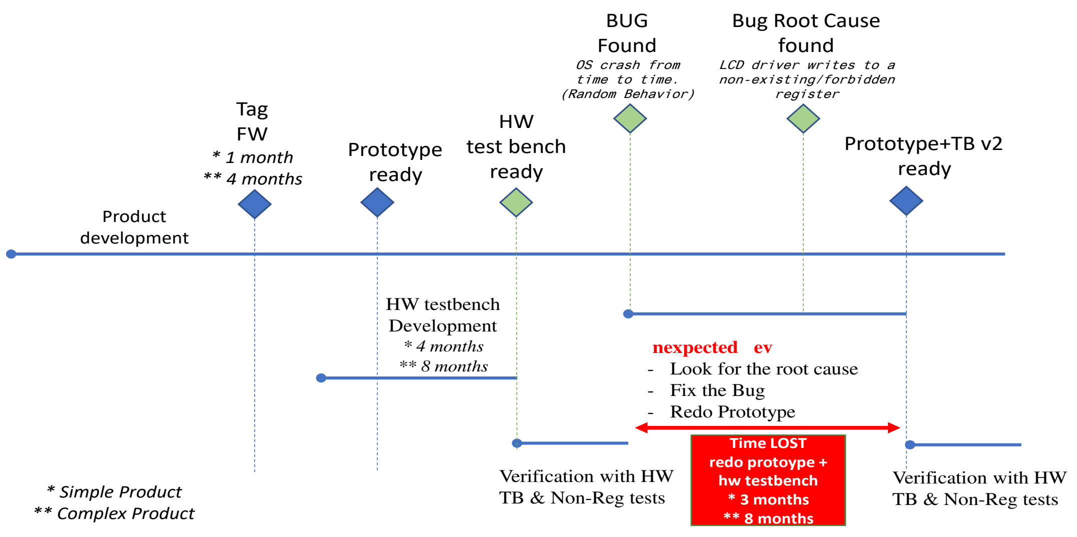


Fig. 8. Time graph if we had not used SystemC-TLM VP

For more complex products such as "connected breakers", containing logic selectivity to trip at times defined by constraining norms, the estimation is that the process will save up to 1 year, partially because of the success of being able to find critical bugs early in the design cycle.

IV. Limitations

Despite the time saved during the development, there are limitations inherited from SytemC-TLM Virtual Platforms.

As our SystemC-TLM virtual platforms are at an LT level of abstraction, and therefore fast, they are not timing accurate. They are not able to be used to find timing issues. Timing constraints are also very important and should be validated using physical devices or Approximately Timed simulators.

Another limitation in this procedure, especially for testers, is that the tests do not cover electrical tests, mechanical tests or electrodynamic tests, all of which are also part of the validation of a full product. One possibility would be to use SystemC-AMS to simulate the analog parts of the design. This is left as future work.

V. Conclusion

While the use of virtual platforms is a well understood technology which enables a shift left in terms of development flow, the application to test equipment has been somewhat overlooked. By applying the virtual platform methodology to both the product and the test equipment, this paper shows how the same shift left can benefit the full platform and test equipment development cycle. The paper details the subtly different requirements of the test equipment, and how, using the abstraction layers typical in modern test equipment, the same test code can be re-used both within a virtual environment and in a real physical environment. This has enabled bugs to be discovered earlier. The paper shows how parallel development of both the test sequences and product firmware can be achieved using a VP-P and VP-T. Overall, the paper shows an improvement in terms of time to market, which is measured in terms of months for the examples studied, and could be as much as years in some cases.

References

[1] L. Khan, T. T. Jeong, G. Park, and A. P. Ambler, "A hw/sw co-design methodology: An accurate power efficiency model and design metrics for embedded system," in *2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, May 2009, pp. 3–7.

[2] S. Zedek, C. Escriba, and J. Y. Fourniols, "Dedicated system for structural health monitoring of aircraft hardware system based on v-cycle model," in *2015 IEEE International Symposium on Systems Engineering (ISSE)*, Sept 2015, pp. 179–183.

[3] "Iso/iec/ieee international standard - systems and software engineering – system life cycle processes," *ISO/IEC/IEEE 15288 First edition 2015-05-15*, pp. 1–118, May 2015.

[4] S. Abdennadher and S. A. Shaikh, "Practices in high-speed io testing," in *2016 21th IEEE European Test Symposium (ETS)*, May 2016, pp. 1–8.