2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE OCTOBER 26-27, 2021

A Novel Approach to Functional Test Development and Execution using High-Speed IO

Marcus Schulze Westenhorst, Solution Architect, Advantest, Böblingen, Germany (*marcus.schulze-westenhorst@advantest.com*)

Jörg Simon, Application Engineer, Cadence, München, Germany (*joergs@cadence.com*)

Markus Bücker, Program Manager, Advantest, Böblingen, Germany (*markus.buecker@advantest.com*)

Klaus Dieter Hilliges, Platform Extension Manager, Advantest, Böblingen, Germany (*klaus.hilliges@advantest.com*)

Michael Braun, Product Manager, Advantest, Böblingen, Germany (*michael.braun@advantest.com*)

***Abstract*— The work presented is the outcome of deploying the portable stimulus standard (PSS) to an automated test equipment (ATE). The target is to demonstrate that by combining design verification tools with the capabilities of an ATE it is possible to enable a new methodology for the development of functional test cases and their execution using high-speed IO interfaces on an ATE system. Alongside the coverage increase, time to market (TTM) and test time will be reduced by downloading and debugging a compiled test executable instead of classical vector patterns. Additionally, vector memory can be freed by adding new and dedicated memory for functional test. Adopting the new methodology involves investing in new hardware and software. The gain will be savings in TTM, test time, increased coverage and therefore an increased quality.**

Keywords—portable stimulus; automated test equipment (ATE); functional test; high-speed IO interfaces (HSIO)

I. INTRODUCTION

Portable Stimulus Standard was introduced by Accellera System Initiatives in June 2018 as Portable Test and Stimulus 1.0 followed by PSS 2.0 in April 2021 [17]. PSS is addressing obstacles which have been restricting the functional verification community before. PSS, for example, supports the portability of test content between multiple platforms by providing a single standard way to specify validation intent and behavior [17]. PSS allows to increase productivity and quality of functional validation [18] and enables customers to choose between different PSS tools for their tasks. These advantages helped PSS to become an emerging trend in functional verification.

Automated Test Equipment (ATE) is mainly used during IC production to detect fabrication defects. ATE usage guarantees a low defect level of delivered ICs, which is essential to provide working devices to customers. Millions of produced ICs require ATEs to be optimized, reducing test time by increasing its degree of automation and parallelization. Typically, the test time per device is dominated by scan test patterns created by automated test pattern generation tools (ATPG). The scan test methodology requires dedicated design for test (DFT) hardware implemented in the ICs inserted by Electronic Design Automation (EDA) tools during the design.

By the combination of the presumably two independent worlds of PSS and ATE significant benefits can be realized. In this paper, it will be explored why it makes sense to combine the design verification tools supporting PSS with the capabilities of an ATE into one consistent validation flow. We further explain how we managed to create the flow and which benefits could be achieved by a novel approach to functional test development and execution using high-speed IO.

II. MOTIVATION

A. *Increasing design complexity*

Recent publications about electronic market expectations [1] and long-term future trends [2] indicate that there is a continued need for new product developments especially in the area of mobile phone processors, high performance computing devices, machine learning and network & cloud computing. The demand for new and extended functionality of SoCs will have a continued impact on increasing design complexity. Since the 1970s, a constant increase of the number of transistors integrated on an IC can be observed which led to the formulation of Moore's law [4], [3] which is e.g. enabled by continuously optimized design processes solving the increasing design complexity.

In addition to the higher gate count, more factors lead to increased design complexity shown in [1]. Based on the example of mobile phone evolvement, it is shown that between 2004 and 2022, the technology nodes used for IC design have changed from 130nm to 5nm while advances in packing technology are allowing for an increasing number of integrated chiplets. Within the same period the mobile phone generation evolved from 3G, 3G+ over 4G and 4G+ to 5G each supporting increasing data rates. Furthermore, an increasing number of integrated embedded IP blocks and functions such as cameras, sensors, memories, GPUs and more led to more and more complex SoC design architectures, including multiple integrated CPUs and bus systems.

B. *Increasing importance of functional test for production test*

The increasing design complexity along the vectors of advanced packaging, technology nodes, higher gate count and more complex SoC architectures leads to consequences for the typical test content on an ATE.

Advanced packaging technologies enabled the usage of chiplets which emerged to one of the hottest trend in semiconductor engineering like many buzzing headlines are indicating [5], [6], [7], [8], [9]. One of the reasons for the increased usage of chiplets is increased yield and consequently reduced cost using smaller dies [10]. Assuming a constant defect density on a wafer leads to less defective dies if smaller dies will be produced. But this cost advantage has consequences for the final packaged System-on-Chip (SoC). Interconnects between the chiplets are not bonded to the package boundaries and are therefore not accessible for test. The integrated functionality is likely distributed over different chiplets and cannot be tested on a per die basis. Both reasons are bringing challenges for classical scan-dominated production tests.

New technology nodes add new failure modes like bridges and necks to the classical failure modes such as stuck-at and shorts [11]. The new failure modes are leading to unpredictive coupling [11], [13] between signals and make test engineers look back to functional test and the original intent of the design [11].

The high number of gates shows the limit of ATPG requiring new methodologies for production test, as [12] indicates. The example suggests that a SoC developed in 22nm technology with 2.5 billion integrated transistors tested with a high ATPG coverage rate of 99.5% still has 12.5 million untested transistors.

Additionally, the more and more complex SoC architectures with their multiple integrated CPUs, peripherals, and bus systems in leading-edge devices allow for "concurrent scenarios" to be executed which are extremely difficult to be detected by typical production test patterns [12].

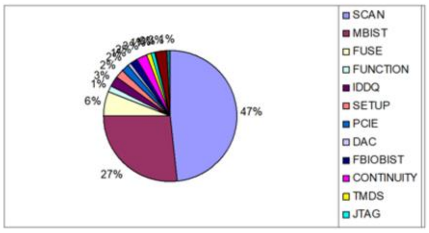

Figure 1. Example of final test time break down.

The typical production test content is currently dominated by structural tests e.g., scan test Figure 1, and will likely see a trend for an increasing amount of classical functional tests due to the points mentioned above.

C. *Resulting opportunity*

Functional test content and typical production test content (mainly structural tests) are showing fundamental differences. Both are usually developed by different teams with different tools. Functional tests are typically developed in the form of embedded software tests and executed in mission mode. In contrast, typical production test content are vector test patterns and are stimulated at the chip boundary with the IC operated in test mode. Functional test content is usually downloaded to the SoCs internal memory via high-speed interfaces using logical protocols. In contrast to this, production test patterns are often stimulating parallel interfaces using physical protocols.

The deltas between functional and structural test content lead to several challenges when functional test content shall be executed on an ATE. Converting the functional test content into typical production test vector pattern requires extensive development time. In addition, on standard ATE equipment, the production test-patterns are stored in the vector memory, a sparse resource on an ATE. This leads to a conflict for using the limited resource of vector memory between scan test and functional test. An increasing number of integrated gates needing more scan pattern will worsen the situation. Finally, test time is one crucial aspect of the production test. Classical ATEs do not support native HSIO interfaces but operate with parallel interfaces in test mode. Taking the given problems into account there will be a novel approach to functional test development and execution using high-speed IO proposed solving for the following problems:

- A seamless SW driven execution of unconverted functional test content on an ATE.
- An ATE HW supporting functional HISO for native instead of physical DUT-ATE communication.
- Bringing functional coverage and constraint random test content to an ATE.
- Providing a native SW debugging environment instead of a vector pattern compare mechanism.
- Enhancing the PSS scenario analysis with a functional viewer integration on ATE.

III. Approach

A. Detailed considerations of selected problems bringing functional test to an execution on an ATE

One question a validation engineer needs to answer is the question about functional coverage which is often hard to be answered and may lead to following approach. First, the validation engineer checks the SoC architecture for potential test paths and potential parameterization needs to access the SoC components. Second, the validation engineer creates out of this analysis validation requirements, maps the test cases to the requirements and reports the number of passing test cases over time versus an expected test case pass curve. It will be shown in the next chapters of this paper how the novel approach will improve the situation for the validation engineer.

Bringing functional test content to an ATE, the validation engineer needs to convert the functional test into a vector-based test pattern which an ATE can execute. A standard methodology to convert functional tests for the usage of an ATE requires quite extensive development time. A complete functional test is usually composed of different sub-tests which are independently simulated during the SoC design phase, such as:

- Initialize the SoC, bring the IC into test mode and map the parallel IF to the chip boundary.
- Stimulate the parallel IF to load the functional test content into the SoC internal memory.
- Switch the SoC into mission mode and execute the functional test.

Historically, the correct bus timing for the parallel interface needs to be applied. During the RTL-simulation the pin level signals need to be strobed and saved e.g. as a VCD file. This VCD file needs to be transformed into an ATE compatible vector pattern format. It will be shown how this pattern conversion step can be avoided.

Bringing functional test content to an ATE requires native software debugging environment on an ATE system to reduce extensive debug cycles. In a native software debugging environment software traces indicate what went wrong during the execution of the test. This helps to generate ideas for potential SW breakpoints. Once identified, an in-system software debugger is started and the embedded SW and register states can be analyzed. On an ATE-based system only the expected and received pattern can be compared and deviations can be observed after the test execution. It is very hard to re-translate the pattern deviation into SW semantics of the original functional test case. Debugging of functional test case failures on an ATE therefore often becomes a joint debugging effort of expert groups including debug pattern creation, pattern re-simulation and pattern re-execution. This process often needs to be repeated multiple times, making functional debug a cumbersome process on an ATE but can be avoided. The following sections of this paper will explain how the process can be improved significantly.

B. Functional test development and execution using high-speed IO

Taking the fundamental problems described above into account the approach bringing functional test content to an ATE-based system needs to be re-thought completely. This includes the test content creation, the interface of the functional test content to the ATE, the ATE SW and HW, and the analysis and debug phase of test content failures. Altogether the considerations led to the development of a novel approach for functional test development and execution using high-speed IO which is visualized in Figure 2 and which will be explained in detail in the next chapters of this paper.

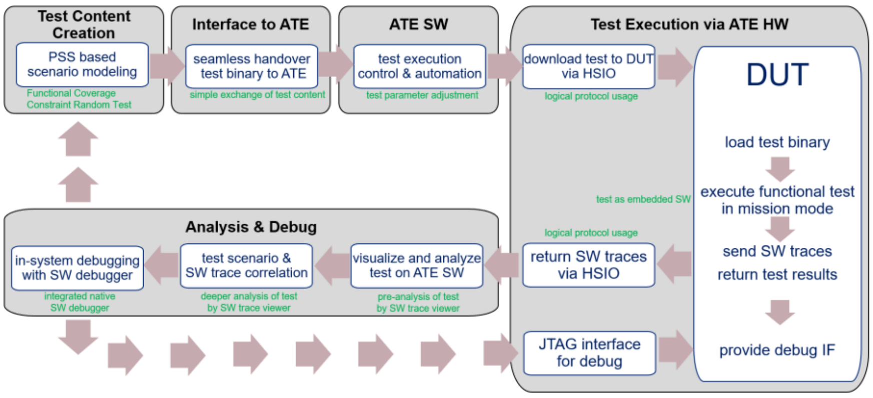
Test Content Creation
PSS based scenario modeling
Functional Coverage
Constraint Random Test
Interface to ATE
seamless handover test binary to ATE
simple exchange of test content
ATE SW
test execution control & automation
test parameter adjustment
Test Execution via ATE HW
download test to DUT via HSIO
logical protocol usage
DUT
load test binary
test as embedded SW
execute functional test in mission mode
logical protocol usage
send SW traces
return test results
return SW traces via HSIO
JTAG interface for debug
provide debug IF
Analysis & Debug
in-system debugging with SW debugger
integrated native SW debugger
test scenario & SW trace correlation
deeper analysis of test by SW trace viewer
visualize and analyze test on ATE SW
pre-analysis of test by SW trace viewer

Figure 2. A Novel Approach to Functional Test Development and Execution using High-Speed IO.

IV. Functional Test Content Creation: using PSS as modelling language

A. *PSS Model Positioning for ATE*

Using the PSS (Portable Test and Stimulus Standard) intends to create for SoCs or sub-systems a rich set of test scenarios in a reusable way. This is achieved by a PSS model written in a DSL (Domain Specific Language) which is defined once with characteristics that it can be shared between multiple projects and teams.

The fundamental reuse aspect of PSS is the base idea for this project. That means, reusing well-defined and portable PSS models in the ATE world that have been created and tested already in the pre-silicon world. Mature pre-silicon verification techniques like coverage and efficient messaging implementations can be adapted as well as technology specific cache coherency or power modeling solutions as part of existing libraries. The PSS modeling capabilities have the flexibility to provide a runtime control from the outside world that enables the integration with an ATE system. Moreover, a solver engine is used to create only valid tests out of the PSS model. Using a PSS model as the fundamental input, verification techniques from the pre-silicon domain is available in the ATE world.

B. *Preparation of a controllable PSS model for Post-Si*

To work with the re-usable PSS model without a long learning curve, a human interface is available. The given and reused PSS model, a UML diagram (compare with the figure [Figure 3] below) with related control knobs will be extracted and proven. The control knobs enable on-the-fly parameterization of the test case on the ATE with its order-of-magnitude faster execution than in pre-silicon simulation or emulation. Leveraging PSS and its tool support even novice PSS users can compose scenarios for ATE, constraint them, and set individual attributes for the PSS model to express the verification intent. For experienced users and in highly automated environments it is possible to define the verification intent in batch mode as well.

Out of the instrumented PSS model, one or more valid scenario solutions will be created. Thus, the PSS model developer has the chance to sort and qualify the tests based on the available generation time coverage, without any required execution of the test itself. Additionally, it is recommended to insert already hooks for the planned ATE integration into the PSS model. These hooks are defined exchange parameters (control-knobs) and return values to influence the runtime behavior of the test and enable analysis.

After all valid test scenarios are identified, C-code will be generated. This code is required to be processed by the appropriate toolchain. This means to execute project-specific compilation, linking and binding steps. The resulting output should be a ready-to-go executable or memory image, depending on the ATE setup. During the execution of that code functional, runtime coverage will be collected. This collected functional PSS coverage is based on constraint random data and represents a major add-on to the ATE world and increases the quality of the Device Under Test (DUT) significantly e.g. by covering defect types described in [II.B]. It is not only based on the classical random data coverage. Moreover, it can and should contain top level scenario coverage, that should be provided by PSS models out of the box. All this together makes the DUT and the related system behavior to become more and more a grey box.

V. Interface to ATE: FDAT interface

An important topic in the ATE world how to manage hundreds or thousands of test executables most efficiently. The question is: How can we easily hand over many tests from one team to another or share it between multiple company locations? The proposal is a kind of bundle or container – it is called FDAT - that contains all the information required to run a single test or a complex tests-suite. The FDAT container (shown as interface in Figure 4] below) is a well-defined infrastructure including the required test binaries and test parameter sets, optional source code, documentation, statistical analysis tools, debug kits, and other items that are required. The container itself is not a fixed format, its open for customized extensions.

VI. ATE SW: FDAT based test execution Control and Automation

A. *FDAT based test execution on ATE*

With single or multiple FDAT containers as an input it is quite easy to step through a complete ATE flow. The ATE engine will be instrumented appropriately, based on the content of the FDAT container. The high-speed IO interfaces to download the test and upload the results are activated appropriately. Potential monitors and debug interfaces are also prepared. The ATE will be started and will communicate via the HSIO interfaces with the DUT. Depending on individual needs, it is possible to work in basically two major modes. One is the interactive mode for debug and or for analysis in a development environment. The second one is the fully automated mode in high-volume production. Regardless of which mode, the explained flow is always the same. To achieve this, new and extended mechanisms are introduced on the ATE side and in PSS modeling. For the ATE HW new elements are developed to enable fast HSIO communication between the ATE HW and the DUT. On the software layer new mechanisms are implemented to control and steer the test case on the DUT via the well-defined PSS interfaces. To read back the internal status of the DUT improved mechanisms for data logging and memory dumps transfers have been developed. And finally, improved analysis-interfaces are available based on the new data streams, which are introduced in the following chapters.

B. *Efficient parameter handling*

One of the newly developed project features is a smart solution to change parameters of the test executed on the DUT, without the need to recompile the complete test. The delivered SW as part of the FDAT container is prepared accordingly, to make this possible. That means there is a major static part of the SW that will be loaded only once for execution. Many small SW pieces represents the parameter sets, that will be exchanged on-demand by the ATE control software - automatically. That enables a high degree of automation for systematic DUT behavior testing, which was not available out of the box before. Additionally, it is guaranteed that those tests are executed with the best possible performance, because the re-load time is reduced to a minimum.

C. *Coverage increase and statistical analysis*

It was explained, how the DUT will be controlled and how the input/output data are processed. Following a brief introduction of the analysis part with the help of an example. Let us assume (as shown in figure Figure 3] below) a multi-core cache coherence use-case was defined with the help of a PSS model. The model contains multiple loops for different parameter sets. The parameter sets itself represents multiple variants for the cache setup. All are encapsulated with the FDAT containers and have been executed already. The provided output data are supplied to the ATE result analysis process. The analysis process will create diagrams, tables and instrumented log files for human inspection and evaluation. Important here is, e.g. the very expressive shmoo diagrams combining two dimensions: The data delivered from the functional PSS model - like the status of cache, mem-size, loops, timestamps etc. - and the data based on physical parameters - like clock and voltage and more. PSS-based run-time coverage is delivered with the functional PSS model. This combination adds a new quality for those types of diagrams and offers a deeper understanding of the internal status of the SoC and dependencies of the executed test.

VII. Test Execution via ATE HW

A. *Smart Compute Instrument*

In this project, a new Smart Compute Instrument (SCI) was developed that was seamlessly integrated into the existing ATE HW & SW. Its key features are the support of native HSIO interfaces, a parallel JTAG interface support and it is equipped with memory. This SCI allows to natively interact with the Device Under Test (DUT) via a low pin count HSIO without the need to physically stimulate a parallel high pin count interface in DUT test mode. The functional test content delivered as FDAT container can be downloaded by the ATE SW directly to the DUT without pattern conversion using the HSIO of the SCI. Because of its memory, the new Smart Compute Instrument can store the functional test content and offload the vector memory of typical digital ATE instruments used for scan patterns. Thus, the conflict of storing functional and structural test content in the limited vector memory could be resolved. Furthermore, the native JTAG interface support offers the opportunity to connect a native in-system software debugger easing the debug process of failing test cases.

B. *Test Scheduler*

To complete the communication chain between ATE SW, the Smart Compute Instrument and the DUT a Test Scheduler was developed which is executed on the DUT. It is the communication interface between the ATE and the functional test case. For instance, the Test Scheduler starts and stops the functional test and downloads the test via the HSIO interface to the DUT memory. Furthermore, it collects software traces and test results generated by the functional test case used during the analysis of test results once the test cases are terminated.

VIII. Analysis and Debug

The collected software traces will be analyzed to debug failing test cases after the set of test execution results (pass/fail overview) has been evaluated. In the newly created software trace view as part of the ATE SW which is called Activity Trace Viewer the validation engineer can evaluate what has happened during the execution of the functional test case on the DUT in a human-readable form. This allows an so far unprecedented view into SoC operations on an ATE for failing functional test cases and was described as one main problem in [III.A]. Furthermore, the so-called Activity Trace can be exported into an external Scenario Viewer and be correlated with the original PSS model. The insights gained by evaluating the Activity Trace SW Viewer and the external PSS Scenario Viewer are likely helping to derive breakpoints for in-system debugging. In this project, a typical in-system debugger for embedded SW applications (Lauterbach Trace 32) was integrated to support embedded SW debugging.

IX. Application

In this chapter, the results generated by implementing the described flow on an FPGA-based DUT and an outlook for real-life application will be shared. The supporting products of Cadence and Advantest will be briefly explained, and an outlook into further applications will be given.

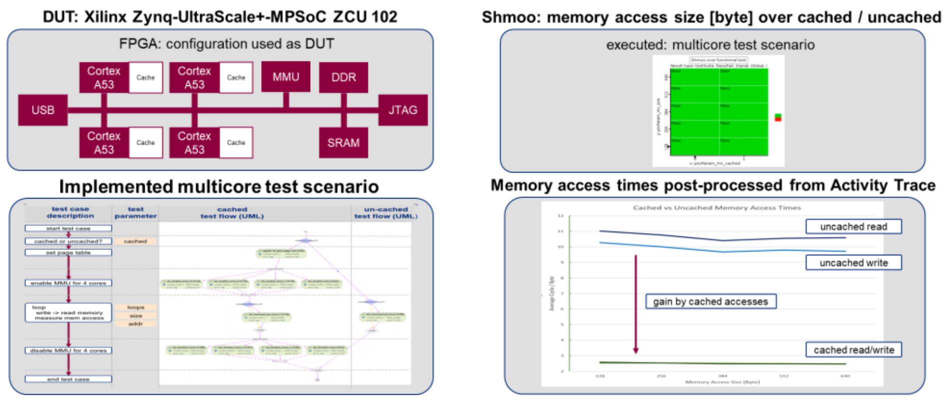
DUT: Xilinx Zynq-UltraScale+-MPSoC ZCU 102
FPGA: configuration used as DUT
Cortex A53
Cache
MMU
DDR
USB
JTAG
SRAM
Shmoo: memory access size [byte] over cached / uncached
executed: multicore test scenario
Implemented multicore test scenario
test case description
test parameter
cached test flow (UML)
un-cached test flow (UML)
start test case
cached or uncached?
cached
set page table
enable MMU for 4 cores
loop write -> read memory measure mem access
loops
size
addr
disable MMU for 4 cores
end test case
Memory access times post-processed from Activity Trace
Cached vs Uncached Memory Access Times
uncached read
uncached write
gain by cached accesses
cached read/write
Memory Access Size [Byte]

Figure 3. Complex devices and use cases are supported. Parameter variation and automation enable design optimization.

A. *Functional Test Cases created by PSS successfully used for design validation on an ATE*

As shown in figure [Figure 3] the project was brought to a state where a complex multicore test case including test parameters was modelled in PSS and successfully executed on a ZCU 102 FPGA configured as depicted in figure [Figure 3]. With the help of the parameter exploration capabilities of the functional test on the ATE a shmoo plot could be created. The parametric results returned as part of the SW traces could be extracted and visualized in a two-dimensional graph. (average cycle per byte access diagrammed over memory access size). This allows for instance to evaluate the SoC specific gain which could be achieved by enabling the cache. This enables so far not possible design exploration capabilities based on functional tests executed on an ATE and was made possible by our novel approach to functional test development and execution using high-speed IO.

B. *Full execution cycle can be achieved within minutes*

The graph shown in [Figure 4] shows the realization of the novel approach to functional test development and execution using high-speed IO interfaces. The tight integration of all components allows executing an entire validation cycle within minutes. A complete validation cycle starts at the test content modification, including recompilation, parameter adaptation, execution of the test on an ATE including result and software trace evaluation, setting up the in-system debugger, stepping through the executed code and comparing with the original PSS model. This is a signification improvement in terms of time and effort compared to the original debugging approach on an ATE as described in [III.A]. The original approach could easily take days to bring the correct expertise together for a joint debug effort and think of the limited availability of ATE resources to re-execute a debug pattern.

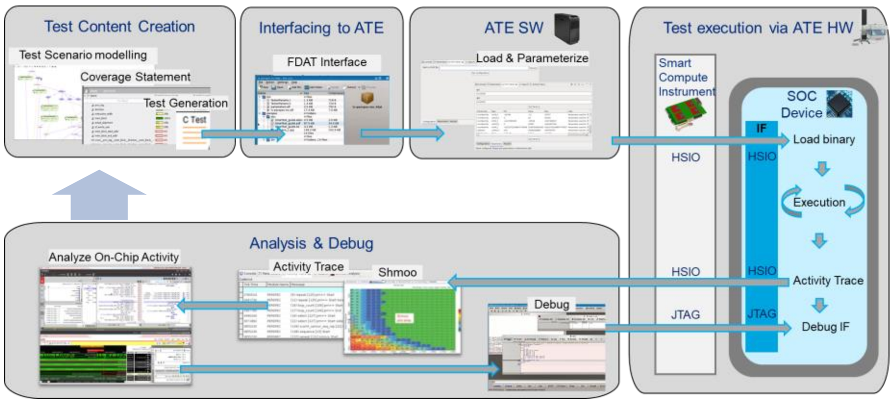

Figure 4. Realization of A Novel Approach to Functional Test Development and Execution using High-Speed IO.

C. *Enabling new ways of work*

Utilizing the developed flow in a real-life application enables entirely new ways of work on an ATE-based system for test content creation, during engineering, and for High-Volume-Manufacturing (HVM). A tool-based functional coverage statement could already be generated at the point of post-silicon test case creation. The coverage of the design can be analyzed by a defined set of parameters which also can be used to change environment conditions such as voltage or temperature to increase the quality of the device further.

The introduced approach allows the usage of a typical SW environment to create, run and analyze test failures while remote access to the ATE is possible as well. During device and test bring-up the usage of known in-system SW debuggers is supported and test failures can be pre-analyzed by the integrated Activity Trace SW trace viewer.

Once the test runs smoothly, the unified SW environment on an ATE allows transferring the test cases easily to characterization or production. In addition, the high automation capabilities of the ATE-based system are enabling a constant monitoring of the device quality based on a complete set of functional tests.

D. *Potential future applications*

The novel approach for functional test development and execution using HSIO interfaces has proven to be working. The validation engineer, therefore, can focus on the test content rather than on the validation environment. Exploring the possibilities of the validation capabilities that the introduced approach could achieve seem to be the next valid step. It is easily possible to extend this approach to address validation tasks like cache coherency testing, bus congestions scenarios, and power measurements during execution or adaptive voltage and frequency scaling. In addition to this it may be evaluated how to simplify environment control options such as temperature even more.

E. Products of Cadence and Advantest supporting the introduced validation flow

- **Cadence Perspec System Verifier** supports Accellera Portable Stimulus Standard (PSS) 2.0, enables correct-by-construction generation of tests, supports easy portability of test content to derivative projects and reports generation and runtime coverage metrics.
- **Advantest SmarTest & V93000** supports collaborative test program development (Java), supports the user with a content assisted UI based on Eclipse (extendable), is a scalable SoC test platform, incorporates innovative per-pin testing capabilities and tests a wide range of devices.
- **Advantest Smart Compute Instrument** is designed for applications like SW-driven functional (self) test and SCAN over standard HSIO IF. It supports JTAG and other control and debug interfaces.
- **Cadence Indago** support synchronous analysis of Activity Trace and source code as well as waveform like analysis of Activity Trace.

X. Conclusion

In this paper, we demonstrated the benefits of a novel approach to functional test development and execution using high-speed IO. This was achieved by combining the benefits of design verification tools with the capabilities of an ATE system. The design coverage can be increased for production test by bringing functional coverage and constraint random testing to the ATE world. The combination of test and environment parameter variation allows for more in-depth design exploration. The functional tests do not occupy vector memory any longer, which further increases structural and functional coverage. The proposed approach is reducing the efforts for functional test development and execution on an ATE significantly by native execution of functional test downloaded via HSIO. Similarly, the native SW bring-up and debug environment introduced significantly reduces the effort for analyzing test failures. Summing up the novel approach to functional test development and execution using high-speed IO improves throughput and Time to Quality (TTQ) while reducing the Time to Market (TTM).

References

[1] Etienne Sicard, Alexandre Boyer, "Impact of Technological Trends and Electromagnetic Compatibility of Integrated Circuits" Impact of Technological Trends and ... (archives-ouvertes.fr) , pp. 2-5, October 2019.

[2] Intel Corporation, "Intel Next 50 Study" Intel-next-50-study-results.pdf, pp. 7-9, August 2018.

[3] Hannah Ritchie and Max Moser, "Moore's Law: The number of transistors on micorchips doubles every two years" Transistor-Count-over-time (ourworldindata.org), p. 1, November 2020.

[4] Gordon E. Moore, "Cramming More Components onto Integrated Circuits" ~/papers/moore.pdf, pp. 1-4, January 1998.

[5] Peter Clarke, "Birth of chiplet market shows more than 40% annual growth" Birth of chiplet ... (eenewsanalog.com), p. 1, May 2020

[6] Mark Lapedus, "Chiplet Momentum Rising" Chiplet Momentum Rising (semiengineering.com), p. 1, February 2020 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] Samuel K. Moore, "3 Ways Chiplets Are Remaking Processors" 3 Ways Chiplets ... (IEEE Spectrum), p. 1, April 2020

[8] Bob O'Donell, "Opinion: The hottest trend in semiconductors is "chiplets"" Opinion: The hottest trend ... (techspot.com), p. 1, July 2019.

[9] Tom Coughlin, "Chiplets For All" Chiplets For All (forbes.com), p. 1, May 2019

[10] WikiChip Chips & Semi, "Chiplet" Chiplet - WikiChip, p. 1, February 2021

[11] Ron Wilson, "SoC testing becomes a challenge" SoC testing becomes a challenge (design-reuse.com), p. 1, Mar 2003

[12] Karthik Ranganathan, Anil Bhalla, "Key trends driving the need for more semiconductor system-level testing" , Key trends driving ... (Evaluation Engineering), p. 1, Jun2 2018

[13] Darayus Adil Patel, "Test and characterization methodologies for advanced technology nodes" Test and characterization ... (archives-ouvertes.fr), p. 9, June 2018

[14] EPS.IEEE.ORG, "Heterogeneous Integration Roadmap / Chapter 17: Test Technology" HIS, Version 1.0 (ieee.org), pp. 44-51, 2019

[15] Lauterbach GmbH, "Debugger Basics - Training" https://www2.lauterbach.com/pdf/training_debugger.pdf, p. 25, 2020

[16] Tom Fitzpatrick, "Portable Stimulus: Standard vs. Tool vs. Language" Portable Stimulus... (siemens.com), p.1, April 2017

[17] Accellera Systems Initiative, "Portable Stimulus Working Group" Portable Stimulus (accellera.org), p. 1., 2021

[18] Accellera Systems Initiative, "Accellera Approves New Portable Test and Stimulus Standard" Accellera... (Accellera), p.1, June 2018