A Novel Approach to Entirely Integrate Virtual Test into Test Development Flow

Ping Lu, Daniel Glaser, Gürkan Uygur, Klaus Helmreich

Chair of Computer Aided Circuit Design
Friedrich-Alexander-University Erlangen-Nuremberg
Erlangen, Germany
{pinglu, glaser, uygur, helmreich}@lrs.eei.uni-erlangen.de

***Abstract* – In this paper, we present an open architecture Virtual Test Environment (VTE) which can be easily integrated into various modularized Automatic Test Systems (ATS) compliant to Open Standard Architecture (OSA). The focus of this paper is to analyze and address the major issues that still prevent the application of Virtual Test (VT) from day-to-day's practice. As a pilot demonstration, a VHDL-AMS based VTE is established and an ADC test is performed. The environment is intended to seamlessly interoperate with the test system during test program development procedure.**

***Keywords* – Virtual Test, Test generation, Simulation, Hardware description language, VHDL, ATML, IEEE1641**

I. Introduction

Virtual Test (VT) [1]-[4], defined as developing and debugging a test with simulation models of the device under test (DUT) – especially for mixed-signal devices – and the associated target test environment, is a technique which has been widely acknowledged as a most promising methodology significantly reducing time-to-market. It allows test development to be accomplished off-line or without a physical device which provides benefits like concurrent engineering and saving expensive ATE time. Moreover, it supplies test engineers with enhanced debug efficiency by full control over the DUT model.

However, though the concept has been introduced over a decade ago, yet it has not achieved industry wide acceptance in test systems as EDA tools have done in the design world. After exploring the historical efforts and attempts of VT, we summarize the following issues causing the gap:

a) Interoperability between VTE and test development system

Virtual test is usually performed in a simulation environment where test equipment, DUT and DI (device interface) are modeled in languages like VHDL/-AMS, SystemC/-AMS/-WMS, etc. Driving these simulation models dynamically to reassemble each test step from the test program or read back responses is a major problem with previous approaches. There are very few implementations (e.g. IMAGE/ExChange with Dantes [2][5]) that achieved some practical impact.

The reason for this is that in former VTEs, only hardware parts of the tester were modeled in the simulation environment, however, in order to run a test process simulation without silicon or even without a tester, its software drivers also need to be covered to interact with the test program from simulation environment.

b) Co-simulation between different simulation environments

In many cases, DUT models coming from designers are written for other simulators than that of the VTE, requiring multiple simulators and co-simulation interfaces. This situation tends to trap the test developer into excessive workload of building interfaces for co-simulation instead of pursuing his native tasks.

c) Modeling effort of tester and fixture

This is the major effort to establish a VTE and it is very hardware dependent. The traditional approach of building a simulation library of everything in a tester with pin and structure compatible degree usually takes several person months. Furthermore, now it fails with the emerging modularized ATE which indicates possible change in hardware during ATE life cycle, and hence would force test developers repeatedly to create models of test instruments from scratch.

d) Simulation efficiency

Though it has been widely recognized that the tester models should be simple enough to save computation time, the legacy ATE systems tend to expose quite some hardware detail to the test engineer and offer drivers on very low layers. This situation avoids tester models being established in high levels of abstraction leading to an unendurable simulation time.

e) Open architecture

Considering the heterogeneous architecture of different legacy ATE systems and their different data formats, chances are very small to construct a platform-independent VTE module which can be easily integrated into different test systems.

A. Motivation

Our research, under the cooperation project OKTOPUS (funded by the German Federal Ministry of Education and Research, support code 01M3182x), attempts to construct a VTE for a modular OSA[6]-compliant rack test system. After considering the historical roadblocks and initial concepts [7], we identify the following key requirements:

a) *Open architecture* – can seamlessly integrated into OSA-compliant test systems. This requires an in-depth understanding of ATS framework, interface boundaries and information sharing mechanisms.

Over the last two decades, a lot of IEEE standards and industrial specifications were proposed (mainly lead by DOD ATS ED [6], SCC20 [8], IVI[9], etc) intending to define and standardize a modular, interoperable, fast developing and easily maintainable modularized automatic test system. Some standards (ABBET, AI-ESTATE, VXI, STD, etc) focus on individual test services by encapsulating critical service modules, defining interface boundaries and data exchange mechanism. Some (ABBET, ATML, OSA), however, focus on providing a common technical framework to facilitate the implementation of ATS architectures based on industry-wide standards interoperability. At present, the widely acknowledged open test system architecture is an information sharing architecture which supports the transfer of information from one life-cycle to another, between components within an ATS and between the ATS and the outside world. The underline objective is to bring together a collection of standard compliant components of ATS with a common information interface; moreover it also grants a new opportunity for an integratable VTE. ATML [10][11] is the most promising candidate. It is defined using XML as exchange medium. With the common format, different tools and systems can exchange information, and form cooperative heterogeneous systems.

'Seamlessly integratable' also indicates to allow sufficient flexibility to cope with different forms of test programs and be able to interact with them by tester models including hardware and software behavior. It requires appropriate data sharing mechanisms to retrieve and feed back information from and to the test system. Further, when conversion utilities are needed, open and friendly interfaces are required.

b) *Less modeling effort* – This has two facets: First, it requires less modeling effort for test instruments. Though we try to offer a VTE which has a wide coverage of test instruments with certain function and structure consistency, still for a modularized test system, there might be scenarios where test engineers or test support teams need to create some of the tester models. The VTE is required to offer facilities to aid the work. Second, it imposes less modeling effort for DUT and DI. DUT models are very test application oriented; one DUT might have different model descriptions for different test cases. This requires simulation environments that inherently can handle different modeling languages and methods as well as good model management facilities.

B. Our Proposal

In this paper, a new VTE is presented to meet the requirements. First, it is an ATML-conformant system, which is able to bidirectionally exchange test-related information (e.g. test data, resource data and historic data) with test systems instead of recreating them. Second, the proposed VTE system offers sufficient flexibility to interact with test programs of different levels. Accordingly, models of test environments are established with different abstraction levels and in such a manner that includes both hardware and software partition. A friendly user interface is built for productively planning the virtual test. Finally, AdvanceMS, a mixed signal simulation environment, is used, supporting a variety of modeling languages including VHDL/-AMS, Verilog/-A(MS), SPICE, C and limited extension of SystemC and MATLAB®/Simulink. Presently, VHDL-AMS and C are our major modeling languages.

II. Modeling of Test Resources

Test resources include a collection of instruments with different functions and interconnects. A test instrument is a physical device and accompanying driver, firmware that is used to accomplish a purpose (e.g. stimuli, measure, switching). In the previous VTE efforts, instruments are tended to be modeled only covering hardware behavior which induce the VTE isolated from the test system and therefore the interoperability is very poor. Nevertheless, this issue is fixed in our VTE by modeling the instruments including both hardware and software partitions. In other words, our instrument models are more conforming to a test programmer's perspective. They are able to interact with driver API compliant interfaces.

Another benefit, other than interoperability, is achieving high level of abstraction. The modern tendency of instrument drivers is targeting at hardware independency; hence the developing, maintaining and re-hosting efforts of test programs would be minimized. The situation leads to instrument class abstraction (e.g. IVI instrument class [12]) and signal abstraction [13] (e.g. IEEE 1641; IVI-SI) for instrument control. Both concepts intend to expose the instruments to the test engineer with their function behavior controlled by common interfaces instead of hardware implementation. This opens the way for us to generate models according to the properties specified by the driver. In our VTE, we support both ways by providing models of signal abstraction and instrument class abstraction.

A. Signal Modeling

Currently, signal oriented test has been widely recognized as the emerging technology in future ATSs. The method grants test programs being written by specifying required signals and desired measurements at DUT pins [14], and likewise, grants the test resource being described with signal capabilities [15]. To support this, ATSs need signal models to define parameters and corresponding behavior, a resource management module to map signal requirements to instrument controls and information models describing capabilities and the connectivity of instruments, switching matrix and fixture. This achieves automatic resource allocation and routing. The associated standard that is most widely acknowledged is known as Signal & Test Definition (STD), developed under the designation of IEEE 1641[16] which provides the means to define and describe signals including interface parameters, underpinned behavior and control interfaces.

In our proposed VTE, a simulation library of Basic Signal Components (BSCs), defined in STD, is built in VHDL-AMS. STD defines BSCs as fundamental, reusable, formally described signal models with more than 60 most basic signal and measurement functions that might conceivable be required on DUT pins. The BSC models are categorized in seven types according to their roles: Source, Conditioner, EventFunction, Sensor, Control, Digital and Connection. More complex signals can be constructed by utilizing composition or hierarchy over the BSCs[16], resulting in a Test Signal Framework (TSF). Later, a GUI entry is introduced to ease the process.

A VHDL-AMS description of BSCs is modeled covering both static and dynamic behavior. The static behavior characterizes the signal with a mathematic expression while the dynamic behavior specifies the BSC‘s internal state (e.g. ‘Active’, ‘Paused’, ‘Stopped’). The state transition of the BSC is controlled by software interface or hardware events, thus establishing synchronization mechanisms (fig. 1). The definition can be found in Annex A, B and C of STD [16]. Namely, the static model is the entity for simulations with the DUT model, and the dynamic model is the key to process instructions from the test program and consequently attain dynamic configurability. The block diagram of a VHDL-AMS implementation is shown in fig. 2. To speed up simulation, the BSC control instructions are modeled with simplified interfaces passing commands or configurations to the models without really modeling the exact bus behavior. Therefore, a top-level test bench in VHDL-AMS can be converted from the signal-based test program easily in a instruction equivalent manner. Further the test sequences can be executed interacting with DUT model through its connected signal models.

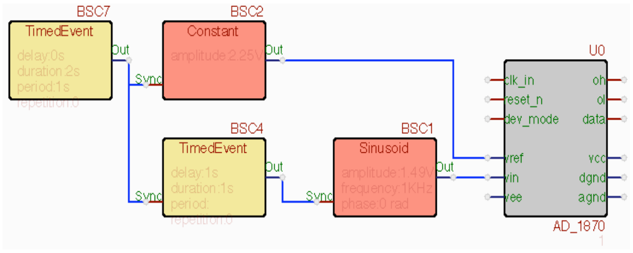

Figure 1. Signal synchronisation example

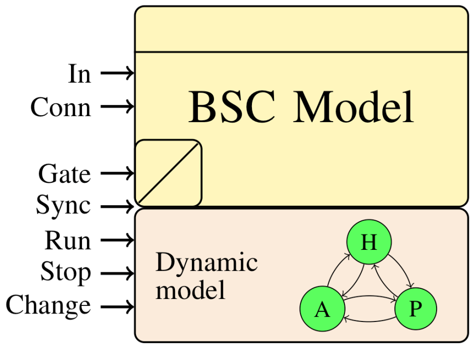

Figure 2. BSC simulation model diagram

B. Instrument Class Modeling

Since instrument oriented test programs are still widely used in practice, a simulation library is built to meet the situation. In order to have a wide coverage of existing instruments, their simulation representatives are modeled compliant to IVI instrument class description. The IVI [9] Foundation develops instrument class standards intending to achieve interchangeability with unification of the concepts, architecture and control interface. It has been considered to be included in a runtime signal interface, bridging between signal requirements and instrument controls. At the moment, eight major instrument classes have been published: IviScope, IviDmm, IviFgen, IviDCPwr, IviSwitch, IviPwrMeter, IviSpecAn, IviRFSigGen.

The IVI-class compliant instrument models are implemented considering both hardware behavior and software interfacing. The hardware part is modeled using behavior-dominant approach to describe the major functions, while keeping the interface consistent – both for data and synchronization timing – with the real I/O port. Software interfacing (represents BUS operations over drivers) is described with simplified bus transaction to pass commands and configurations. Further, a dynamic FSM model is built to manage the internal state transitions in terms of software control or hardware events. All these models are encapsulated in IVI capability classes. An IVI instrument class is composed of several IVI capability classes with most common functionality for an instrument category. The function behavior of an instrument is typically a signal nature, therefore we established a mechanism to describe the instrument capability with the BSC library. An IVI instrument class compliant model, as depicted in fig. 3 with a hierarchical description, has several capability models; each capability model is further composed with a composition of BSC models.

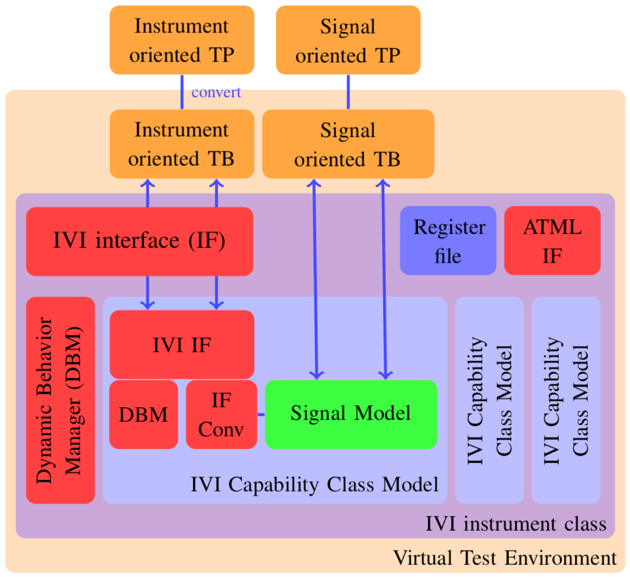

Figure 3. IVI-compliant instrument model diagram

A collection of published IVI class instruments are modeled with this approach. In addition, libraries of BSCs and instruments also supply symbols for schematic entry.

C. Hardware Consistency

Models of ideal functions alone are not adequate for some VTs. More effort need to be done to expose all aspects of the signal integrity at the DUT pins including timing, waveform levels and waveform shapes together with distortion. This is achieved by modeling the instrument with realistic behavior.

The instrument imperfections are divided into two parts: First, static nonlinear behavior (e.g. range, resolution, precision, linearity) and statistical phenomena – this can be encapsulated in the signal model and parameters can be easily measured or retrieved from specification. In fact, an IEEE 1641 based description of instrument capabilities with certain imperfection is defined in [15] with ATML instrument description files. Different instruments will have different ATML instance files. This information is used, in ATS, to verify and choose instruments according to signal requirements; however in our ATE, it is used to bridge generic BSC models or IVI-class models to particular instrument imperfections. During simulation, the models will address the specific ATML instance and load imperfection parameters autonomously.

Another part is the electrical interface. It is a complex and difficult task to build the complete and accurate model even with today's technology not to speak of its expensive simulation cost. In practice, a model with adequate accuracy will obtain more acceptances. It more or less depends on the application and simulation requirements, in [5] an example is shown with a digital I/O stage. The electrical interface is modeled with primitive components like resistor, capacitor, inductor and semiconductor circuit components, etc. Our VTE provides models in different levels of accuracy, so that test engineers are able to make fast and proper choice between efficiency and accuracy. These models are built through characterization of existing instruments through measurements and calculations (e.g. TDR techniques).

Therefore, the full model of a specific instrument will be a combination of BSC models determining its function behavior with some static nonlinearity and noise; and an electrical interface which is alternative in terms of simulation accuracy.

Additionally, signal paths from test system to DUT pins are also in charge of the signal deterioration. The environment provides several models of electrical interfaces and wires with different levels of accuracy to ease the design and verification of DI with minimized modeling efforts.

III. Schematic Entry

In order to run a simulation with a model of the DUT, an explicit test setup is necessary other than a test program. Accordingly a schematic entry will help greatly to enable engineers to graphically construct the test. The graphic entry, named "TPConstructor", is developed with Tcl/Tk based on ATML framework. Right now, it can interact with text formats including ATML and VHDL-AMS. The symbols are in SVG format. TPConstructor is capable of parsing relevant ATML or VHDL-AMS files for necessary information of models and assigning them with automatically generated SVG symbols, if there is no manually designed symbol already associated. Afterwards the simulation models can be manipulated (setup/configure) through their graph representatives efficiently and easily.

For VT, it helps to generate test benches in two diverse ways: signal based test setup and instrument based test setup. For signal based test, signal requirements are constructed visually on canvas with relevant BSCs and then connected to the DUT symbol (fig. 4). A test platform-independent ATML test description file is generated corresponding to the graphical description. Further edit might be needed for completeness and the resulting file may be re-hosted in any other ATML based ATS. In our environment, a VHDL-AMS test bench is able to be generated alternatively from the ATML file or directly from the canvas. Likewise, an instrument based test bench can be created quickly, however using instrument models (fig. 5). Besides, an associated script file is automatically generated to interact with the simulation environment (AdvanceMS) running in background.

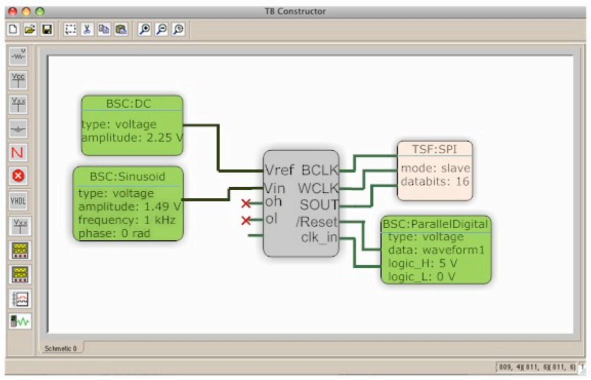

Figure 4. Signal based test construction

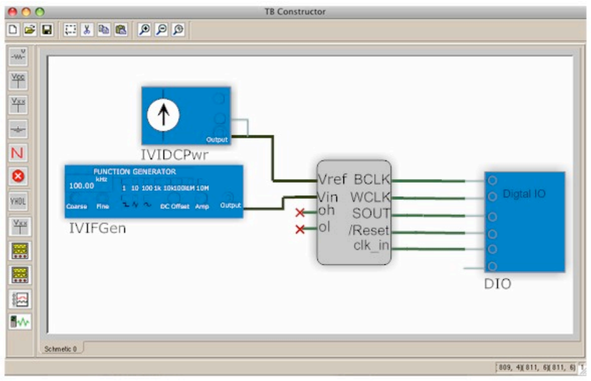

Figure 5. Instrument based test construction

Potentially, the tool also guides the engineer during SOC verification process to generate the test bench and run simulations as easy and fast as any EDA tools do.

With the steps described above, an ATML based integratable VTE is accomplished, allowing test engineers to focus more on developing and debugging test programs or verifying the DUT, instead of modeling the environment.

IV. Modeling of DUT

Far before silicon is available, virtual test can be performed with tester models and DUT models which are delivered by designers in a refinement manner, according to the design process. Hence one DUT might have different model descriptions, considering that test engineers also generate DUT models from scratch for certain test purpose. It is important that the DUT model which is put in a test shall meet the test purpose and in the meanwhile expose only appropriate detail for simulation performance. In this work, a Y-chart [17][18] based approach is developed to inspire test engineers to make a fast decision on the suitable model, individually or combined, for a particular test while taking into account DUT‘s structure and application.

As a test case, an existing product, AD1870[19], is used to demonstrate VT. AD1870 is a stereo, 16-bit oversampling ADC based on sigma-delta technology. Over 20 block models are associated to describe the ADC in different abstraction levels or views. Using the Y-chart based approach, the hierarchical combination of the models are evaluated before each test for yielding an appropriate accuracy and efficiency. For comparison, a test board is built for real measurements. Experience gained so far shows high correlation of real an virtual test.

V. Conclusion and Outlook

The lack of open standard interoperability has been one major reason preventing VT from use in commercial practice. In order to fill the gap, an ATML framework based VTE, consisting of an extendable simulation library of test resources and a friendly user interface is presented. The objective is, by means of our platform independent VTE, to allow the test development process to run efficiently without a physical DUT device or test system. The major contribution of this paper is introducing the Open Standard Approach into VT technology: Formalized resource description – ATS-specific information is encapsulated in a common intermediate format (ATML) facilitating to achieve a test platform independent VTE. Formalized resource control – test resources are modeled with standard-based control interface (STD, IVI-Class) to achieve interoperability. Formalized test description – a graphical assistant tool is developed to aid tests being constructed with standard-compliant common instruments (IVI-Class) or common signal requirements (STD) and in the end generate a test description file in ATML.

To be accepted by the industry, more efforts are required to enhance the total integration for various test systems. At present, the models for test resources, programmed with VHDL-AMS and C, have some deficiencies caused by the language inherent constraints, not being abstract enough for system modeling and not being capable of interacting directly with test programs. Furthermore, it relies on a commercial simulation environment. We intend to solve the problem by adopting SystemC [20][21] and its extensions (-AMS/-WMS) as our major simulation environment. For that reason we are currently migrating to SystemC. It is expected to boost the integration in following way: first, it supports modeling in a wide abstraction level, from specification down to RTL level, with SystemC-AMS even to primitive (electrical) component level. It is especially apt for describing complex systems while offering significant productivity. Second, being able to describe both hardware and software partition in C++ language makes it a perfect candidate for instrument modeling including its driver set. Further, we are looking for means to integrate the SystemC model in instrument drivers as an enhanced simulation feature. The simulation kernels are class libraries, which facilitates SystemC to be integrated into a VTE.

Besides, a lot of enhancement features such as distributed simulation [22][23] on different hosts, co-simulation with other modeling languages and graphical modeling are being studied widely with some pilot proposals.

Further improvements on simulation efficiency are also being examined in our project, including modeling algorithm, distributed simulation, other models of computation [24][25] and potential hardware acceleration.

REFERENCES

[1] Helmreich, K.; Reinwardt, G., "Virtual test of noise and jitter parameters," Test Conference, 1996. Proceedings., International, pp.461-470, 20-25 Oct 1996

[2] Einwich, K.; Krampl, G.; Hoppenstock, R.; Koutsandreas, P.; Sattler, S., “A multi-level modeling approach rendering virtual test engineering (VTE) economically viable for highly complex telecom circuits,” Design, Automation & Test in Europe Conference & Exhibition, 1999. DATE '99 User’s Forum, pp.227-231, 1999

[3] Perkins, E.G.; Wong, J.J., "VTest program mixed-signal virtual test approach," AUTOTESTCON, 97. 1997 IEEE Autotestcon Proceedings, pp.60-71, 22-25 Sep 1997

[4] Miegler, M.; Wolz, W., "Development of test programs in a virtual test environment," VLSI Test Symposium, 1996., Proceedings of 14th, pp.99-103, 28 Apr-1 May 1996

[5] Jean Qincui Xia; Austin, T.; Khouzam, N., "Dynamic test emulation for EDA-based mixed-signal test development automation," Test Conference, 1995. Proceedings., International, pp.761-770, 21-25 Oct 1995

[6] http://www.acq.osd.mil/ats/

[7] Lu P., Helmreich K., “Standard-Based Construction of a Virtual Test Environment”, 20. ITG/GI/GMM Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen" Proceedings, pp.19-23, Vienna, 24-26 Feb 2008

[8] http://grouper.ieee.org/groups/scc20/

[9] http://www.ivifoundation.org/

[10] http://grouper.ieee.org/groups/scc20/tii/

[11] Malesich, M., "Advances in DoD'S ATS framework," Autotestcon, 2007 IEEE, pp.57-63, 17-20 Sept. 2007

[12] Bode, F., "IVI comes of age: an overview of IVI specifications with current status," AUTOTESTCON Proceedings, 2002. IEEE, pp. 317-323, 2002

[13] Ramachandran, N.; Oblad, R.P.; Neag, I.A.; Tyler, D.F., "The role of a signal interface in supporting instrument interchangeability," AUTOTESTCON Proceedings, 2000 IEEE, pp.403-416, 2000

[14] Cornish, M.; Brown, M., "Implementing IEEE 1641 - a demonstration of portability," Autotestcon, 2005. IEEE, pp.144-152, 26-29 Sept. 2005

[15] “Capabilities in ATML” Draft Version 0.32, Capabilities Working Group of ATML

[16] "IEEE Guide for the Use of IEEE Std 1641, Standard for Signal and Test Definition," IEEE Std 1641.1-2006, pp.c1-191, April 30 2007

[17] A. Kienhuis, "Design Space Exploration of Stream-Based Dataflow Architectures: Method and Tools," Ph.D. Thesis, Delft University of Technology, 1999

[18] Waxman, R.; Bergé, J.-M.; Levia, O.; Rouillard, J.; "High-Level System Modeling: Specification and Design Methodologies",1996

[19] http://www.analog.com/

[20] http://www.systemc.org/home/

[21] Martin, G., "SystemC and the future of design languages: opportunities for users and research," Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on, pp. 61-62, 8-11 Sept. 2003

[22] Kai Huang; Bacivarov, I.; Hugelshofer, F.; Thiele, L., "Scalably distributed SystemC simulation for embedded applications," Industrial Embedded Systems, 2008. SIES 2008. International Symposium on, pp.271-274, 11-13 June 2008

[23] Amory, A.; Moraes, F.; Oliveira, L.; Calazans, N.; Hessel, F., "A heterogeneous and distributed co-simulation environment [hardware/software]," Integrated Circuits and Systems Design, 2002. Proceedings. 15th Symposium on, pp.115-120, 2002

[24] Vachoux, A.; Grimm, C.; Einwich, K., "Extending SystemC to support mixed discrete-continuous system modeling and simulation," Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on , vol. 5, pp. 5166-5169, 23-26 May 2005

[25] Patel, H.D.; Shukla, S.K., "Towards a heterogeneous simulation kernel for system-level models: a SystemC kernel for synchronous data flow models," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol. 24, no. 8, pp.1261-1271, Aug. 2005