Development of Test Programs in a Virtual Test Environment

M. Miegler, W.Wolz
University of Erlangen-Nuernberg
Institute of Computer-Aided Circuit Design
Prof. Dr.–Ing. W. H. Glauert
Cauerstr. 6, D-91058 Erlangen, Germany

Abstract

An environment for the efficient development of quality-assured mixed-signal test programs is introduced. The new approach provides links between design and test engineers based on a standard test description language VTML (Virtual Test Modelling Language). The language provides standardized description models for test system resources which can be mapped as well to equivalent simulation models as to real world test system hardware. Methods are provided to check the data consistency of test programs and to validate test program behavior using simulation models.

Introduction

The development of CAD tools for integrated circuit design has undergone dramatic progress during the last years. Although most CAE tool vendors offer a full suite of Design-for-Testability modules (automatic scan path insertion, simulation-to-test waveform converters etc.), there is still a gap in computer-assistance between test pattern generation and test program implementation. After test patterns and test signals have been defined on a higher level of abstraction, real waveforms must be generated by the testsystem hardware and brought to the I/O pin of the device under test. For that purpose a lot of tedious handwork must be done: Various test instrument parameters must be set. Clocks must be assigned and triggers must be defined to synchronize instruments. In order to minimize such efforts and to avoid erroneous test setups without the use of expensive testsystem access, a virtual test development environment is mandatory.

By today, several CAE tool vendors have already addressed the problem of test program generation and verification in a virtual development environment [Cad95]. On the other side ATE manucfacturers became aware of the problem and offer specific solutions to transport simulation data to their systems and to verify test program behaviour [Ter93, LTX93, Ter95]. Meanwhile standardisation efforts are in progress which lead to several IEEE proposals [IEE92, IEE95]. But still, there are subtle problems in establishing an acceptable level of usability for this kind of virtual tools, because of the lack of generalized, system-independant test resource description models.

In this paper, a test development environment for the generation of mixed-signal test programs will be introduced based on a high-level description language: the Virtual Test Modeling Language VTML. VTML-descriptions are processed by C++-modules from a Virtual Testbench Library VTB-LIB. Furthermore, the new language is designed in such a manner that all VTML descriptions can be translated easily and automatically into VHDL-A simulator code of a virtual testsystem.[1]

Test program development

There are three major drawbacks of on-line test program debug using a real-world testsystem: first, this method is very cost-intensive, (2) there is only limited access to an engineering or even production floor testsystem, and (3) both a real device and a loadboard must be available at the time of program debug. These conditions put much pressure onto the test engineers and - in many cases - still lead to a delay in test program acceptance.

Therefore, tools should be available that allow for the generation, verification and debug of test signals and full test programs without access to real-word devices like IC prototypes and testsystem hardware. Using a VIRTUAL (simulation-based) test development environment, all the bottlenecks mentioned above can be circumvented.

In a virtual environment, of course, models must be available for all modules, i.e. a DUT model must be at hand as well as models for all hardware modules of the target testsystem. If these models are available, all aspects of waveform integrity at the DUT pins like timing and waveform levels and waveform shapes may be observed and checked out against the specified or expected values.

It is a major concern of using virtual instruments that restrictions of the real-world hardware are taken into account within the description (VTML) and the simulation models, which are generated out of VTML.

1 The development contributes to the JESSI project AC-6 'Test Development and Design-for-Testability Support'. For testsignal description part of VTML the language TSDL (Test Signal Description Language) has been developed in cooperation with Siemens EZM, Villach (Austria) and the FZI Forschungszentrum Informatik Karlsruhe (Germany). This project is founded by Bundesministerium für Bildung und Forschung and Siemens.

0-8186-7304-4/96 $05.00 © 1996 IEEE

Therefore, the process of test resource modelling is one of the most challenging tasks if analyses shall be meaningful and predictions shall correlate with the real-world behavior later-on. A virtual test bench VTB (fig.1) has been developed that provides a standardized test development environment.

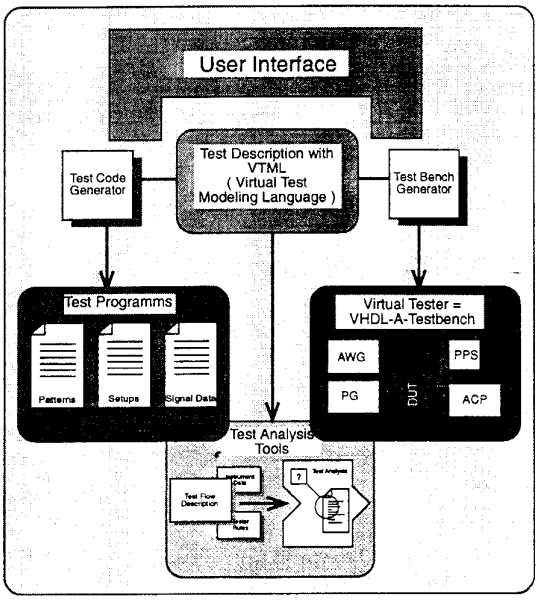

Fig. 1: Virtual Test Bench VTB

The Virtual Test Bench VTB

VTB, a standardized test development environment, provides a common interface to both the design and test engineer: Design and test verification are based on a virtual tester 'programmed' in VTML. Because a virtual tester always reflects the architecture of real-world test equipment, a simulation-based virtual testsystem can be mapped onto a real-world testsystem in a very efficient way. VTB provides a means for checking out test descriptions for consistency using rule-based tools, i.e. even before test programs are evaluated in a simulation-based environment. These early checks allow for the fast and efficient modification of test descriptions such that simulation-based analyses start from a consistent data set.

The Virtual Test Bench VTB is based on a library VTB-LIB, a basic library for VTB tools.

VTB-LIB - the library

VTB-LIB is a library of C++-classes managing all data objects needed for the description of mixed-signal tests. All tools that use the VTB-LIB functions have access to the following features:

- programming of Virtual Test Instruments
- handling of analog and digital signals
- handling of variables
- creation of pin lists and modeling of pin properties
- synchronization of signals and instruments
- generation of test setups
- test-flow description
- common language interface
- data analysis (consistency check, error check)

Tools using VTB-LIB can exchange data amongst each other. There are three different paths for data transfer:

- File transfer: All VTB descriptions are user readable and in ASCII format (read & write operations).
- Serial data stream: For fast data transfer between VTB tools, a standardized serial data stream format is provided. No interpretation of data is neccessary.
- Direct access: Data access through C-functions is available for retrieval, modification and deletion of data objects (provided by VTB-LIB).

Many data access and service functions are available for tools based on VTB-LIB and therefore, test code generators or VHDL-A model generators can easily be implemented. For programmers, VTB-LIB provides a standard interface for data manipulation. Data can be brought into the system by means of editors, text shells or graphical user interfaces. Because VTB data models already take care of the architecture of real-world test equipment, all high-level descriptions can be translated into simulator and testsystem languages very easily.

The Structure of VTB-LIB

The structure of VTB-LIB and its data objects is sketched in fig. 2.

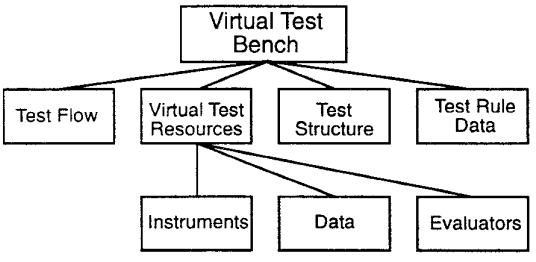

Fig. 2: VTL-LIB data objects

VTB-LIB is based on object-oriented data models for a system-independent description of mixed-signal tests. VTB-LIB data are structured hierarchically. Related data classes are bundled (superior classes). VTML-descriptions are grouped into three major data classes: (1) test flow, (2) test structure and (3) virtual test resources (VTRs).

The "test flow" describes a sequence of test commands that can be applied to Virtual Test

Resources. Test flow sequences can depend on each other.

The "test structure" describes how individual test resources communicate to other test resources and to the device under test. Data sources can be assigned to "Virtual Instruments", and event channels can be defined for trigger control and synchronisation of instruments.

All tests are constructed from Virtual Test Resources that are the elementary components of the Virtual Tester (e.g. signal generators, pin lists, patterns etc.). Although all VTRs are testsystem-independant, they are designed according to the requirements of real-world test equipment.

During test program design, data objects are defined individually and linked together in order to achieve clear descriptions of complex tests. Libraries of reusable data objects may be created (e.g. pin lists). Details of some VTB test resources are described in the following.

Data Resources

Data resources are objects providing data for "Virtual Instruments". Data resources can be waveforms, patterns, pin properties, pins lists etc. Data resources are assigned to instruments within the test structure description.

Virtual Instruments

```
set resource/instrument/signal_source/AWG:demo_AWG
{
    % set up for arbitrary waveform generator
    clock % define clock ratios
    {
        clockdiv := 7680;
        clockmult := 1;
    }
    signal_param
    {
        config
        {
            amp := v0dbm0; % ser variable v0dbm0
            offset := 0V; % Signal offset
            filt := 4KHz; % reconstruction filter
            mode = 'cont'; % continuous wave form
        }
        constant
        {
            bits := 12; % bit resolution
            snr := 60dB; % SNR
        }
    }
}
```

Fig. 3 : Instance of an instrument in VTML

Virtual instruments (VTRs) represent models for signal generation and capturing. Sophisticated 'high level'-instruments are planned, which support standard test methods (DC tests, AC tests, linearity test for ADCs etc.). VTRs are characterized by a set of commands (how they behave), their control functions (how they can be controlled), interfaces (to other instruments), the instrument's clock, and the parameters for an instrument's action.

- Commands: Each instrument is designed to perform certain actions. These actions are defined within the instrument's command vocabulary. (i.e. 'Force Current', 'Measure Voltage' etc.).
- Control Functions: Commands are activated and deactivated by control functions which are sent to the instruments by a "control sequence" (e.g. start(), stop(), continue(), wait() etc.).
- Interfaces: Each instrument is equipped with standard interfaces for their communication to other VTRs.
- Instrument clock processing: Each instrument can be synchronized to external master clocks.
- Parameters: Each instance of an instrument (object) is characterized by a set of parameters which define one single instrument setup. Parameters can be numeric values or VTB variables (defined elsewhere). Fig. 3 shows an Arbitrary Waveform Generator (AWG) using VTML syntax.[2]

Evaluators

Analysis and post-processing of analog signals (e.g. FFTing the data) is described using "evaluators". Each evaluator object is equipped with a list of DSP functions and signal descriptors which refer to signal data objects.

Test Analysis Tools of VTB

Test analysis tools are used to support the engineer during the development of test programs. Because of the growing complexity of test programs, manual optimization is a tedious task. Applying computer-assisted test analysis tools shows test program deficiencies and probably weak points immediately and therefore, optimizations can be performed early in the development process (in most cases, this should save of a much debug time). Test analysis strategies are based on the following assumptions:

- Use as few testsystem resources as possible.

  Because each test system provides only a certain amount of resources, it is good style to keep the number of used resources low (using many instruments makes setup and data transfer more complicated than necessary; but see below: applying many instruments concurrently can lead to decreased testing time).
- Operate instruments concurrently.

  Applying instruments concurrently can shorten test time. Whenever measurements and/or post-processing of the captured data can be interleaved, overall test time can be reduced.
- Optimize test flow with respect to data transfer.

  Measurements should be organized in such a manner that the number of data transfer operations can be minimized.
- Minimize number of instrument setup phases.

  Each initialization of an instrument is time-consuming. Therefore, as many measurements as possible should be carried out with an minimum of instrument setups.
- Avoid redundant commands.

2 A comprehensive collection of Virtual Instruments is under construction at the time of writing.

Test programs consist of many modules. If modules are exchanged, removed or added, some commands may be called twice (e.g. pin setups).

- Avoid programming of delays.

  Delays are often inserted to be sure that all actions are finished. This idle time should be used for useful actions such as setting-up an instrument.

```
if(<proc1>=='ready' && <proc2> == 'finished')
     cmd1;
     cmd2;
     ...
else
     cmd1;
     cmd2;
     ...
```

Fig. 4: Test flow as a set of logical dependencies

Test analysis and optimization tools refer to a test flow description. The final sequence of commands is not yet established but short threads of commands are tied into a network of logical dependencies. The dependencies of these threads are formulated as nested "if-then-else" blocks thus enabling tools to analyze several flow variants and allowing them to find an optimal sequence of commands.

Test analysis requires additional data about specific habits and the behavior of real-world tester resources in order to find optimal solutions for a given test system. This information must be supplied by the test engineer in the form of data files. The following information is required: (1) availability and (2) timing behavior of testsystem resources.

Availability: Which kind and how many instruments are available ? Can they be used concurrently ?

Timing behavior: How long last setup and data transfer phases and which durations depend on other test operations or test program parameters? (e.g. loading of a waveform memory depends on the number of samples).

Test analysis tools show the following information:

- duration estimations for several test flow paths,
- occupancy of test resources during testing,
- revelation of dead locks,
- optimization of instrument occupancy, whereas availability and timing behavior are taken into consideration.

Simulation-based Virtual Testsystems

Although helpful during test planning, estimations based on a formal test flow description cannot show all details of the real-world behavior of the test. Therefore, the test program must be evaluated either on real-world equipment or using a simulation-based "Virtual Tester". Simulation-based evaluation benefits from the fact that the verification can be performed before first silicon is available. Simulation shows whether (1) the response of the device under test is correct, (2) the test equipment generates proper signals and (3) the instruments perform correct measurements at the right time. For that purpose VTML descriptions - which are the 'program' for the Virtual Tester - have to be mapped onto appropriate simulator models (currently, VHDL-A models are under development for faster simulation).

The modeling of real-world test equipment is a time consuming task: For each instrument a simulation model must be created. To reduce the implementation effort for simulation models, we define three classes of models: (1) system control models, (2) standard component models and (3) testsystem-specific models. This ordering is based on the assumption that inaccuracies of real-world tester component (e.g. noise, non-linearity, jitter etc.) are independent of tester control logic. The control logic may only introduce timing 'faults'.

Therefore, we introduced a standard test system control model which is represented by a purely digital VHDL model. This model is supposed to be able to emulate the manufacturer's specific control logic (event scheduler, synchronizer, trigger etc.). To adapt this model to the behavior of real-world test equipment, the timing of the controller model can be varied.

On the other hand, hardware-specific behavior of digital pin electronics, analog signal generators (e.g. direct digital synthesis, arbitrary waveform generator), analog capture ports etc. will be simulated by either analog behavioral VHDL-A models (which can be customized by a set a of parameters) or circuit-level models of specific hardware components.

The common system control and behavioral models will be stored in a standard tester model library: VTB-SIM. All other models must be tailored according to hardware data sheets and circuit diagrams.

The Virtual Tester will be generated from VTML descriptions by a simulator conversion tool which will build-up the virtual instruments using VTB-SIM and the specific models. Fig. 5 shows the structure of a virtual test instrument model. Analog components are marked-up with a bold outline. The rest of the model consists of purely digital system control. The analog part of the instrument model is quite small such that shorter simulation time can be predicted.

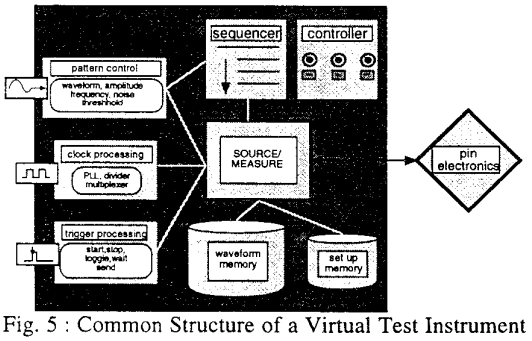

Fig. 5 : Common Structure of a Virtual Test Instrument

Conclusions

A virtual test development environment VTB has been introduced that allows for the system-independent definition of analog, digital, and mixed-signal tests using standardized instrument and data models. The

objective is to achieve short development durations for quality-assured test programs. The following VTB-based tools are already available at the time of this publication: signal generation modules, rule-based consistency check modules, import functions for foreign data formats. The development of a graphical user interface has been started. Instrument models for the Virtual Tester have been developed and are implemented in HDL-A. A preliminary version of a VTB implementation (TSDL, see footnote #1) for Siemens AG is current being released and will be evaluated using real world test program examples.

Acknowledgement

The authors would like to thank Mr. G. Krampl at Siemens EZM Villach (Austria) for his strong support of the development and his contributions to the concept, and Prof. K. Müller-Glaser, FZI Karlsruhe (Germany), who initiated and still supports the development of CAD-integrated CAT tools. Furthermore thanks to Prof. W. Glauert for his constructive criticism and ideas.

References

[Arn92] R. Arnold, M. Chowanetz, W. Wolz, K. D.Müller-Glaser:"Test/AGENT: CAD-Integrated Automatic Generation of Test Programs",
IEEE International Test Conference 1992, pp 854-859

[Cad95] "Cadence Virtual Test Backgrounder", product information, Cadence Design System, 1995

[Kra91] G. Krampl: "Integration von Design und Test mittels höherer Mustersprachen", mikroeletronik 4/1991, pp. 166 - 168

[Cau95] Pascal Caunegre, Claude Abraham, "Achieving Simulation-Based Test Program Verification and Fault Simulation Capabilities for Mixed-Signal Systems" SIEMENS AUTOMOTIVE,IEEE95, p.469ff.

[Ter93] "Creating a Mixed-Signal Simulation Capability for Concurrent IC-Design and Test Program Development", Teradyne Inc., ITC 1993

[LTX93] "Tools and Techniques for Converting Simulation Models into Test Patterns", LTX Inc., ITC 1993

[Ter95] "IMAGE ExChange: An Enabling Technology for Virtual Test", Teradyne Inc.

[IEE92] "TRSL Requirements and Specification", IEEE PAR 1029.3, February 1992

[[IEE95] "Standard Tester Interface Language - STIL", Proposed IEEE Standard P1450 - Review Document, September 1995

[ROA94] 'The National Technology Roadmap for Semiconductors', San Jose, USA, 1994