Analog Behavior Refinement in System Centric Modeling

Author 1 Author 2 Author 3

Affiliation
Organization
Address
emails

ABSTRACT

SoC designs consisting of analog, digital, mixed signal, RF and software blocks are commonplace. SystemC AMS offers the potential for a unified modeling approach for such systems through executable specification. SystemC AMS uses different Models of Computation to gain simulation performance, most notably by abstracting timing information which demands the simulator solvers to be fast, fair and simple.

In this paper we present a method to fortify SystemC AMS extensions with commercial analog solvers while maintaining adequate simulation speed for an overall system simulation. These solvers are more exact to describe specific circuit behavior in nonlinear conditions and wide time ranges. An example system is also presented that implements the proposed scheme.

1. INTRODUCTION

In an increased complexity of heterogeneous embedded systems, system level design through system level languages is a requisite. Whereas system level languages e.g. SystemVerilog offer elevated visibility into the system, they cannot sufficiently point particular behavior (critical analog/RF behavior and nonidealities) that has to be understood for design drivers and product requirements. Therefore system level simulation should be augmented with corner views of design areas that are not always discernible.

High abstraction and system level AMS modeling was a natural sequel after Open SystemC Initiative thoroughly addressed the digital domain of design and still presses on. The analog and mixed signal extensions to SystemC [3] tritely known as SystemC AMS have been formally released as a first-cut draft proposal. The current SystemC AMS prototypes offers three Models of Computation (MoC): Electrical linear networks and linear signal flow (transfer function, pole-zero or state space representation of input/output behavior). Both modeling paradigms embed in SystemC `sc_method()` class. The third is synchronous Timed Data Flow (TDF) MoC with indigenous `processing()` method which is a solver for computing the continuous time behavior of the model as defined by the user. Both linear MoCs solve linear implicit differential equations [13] at appropriate time. Simple nonlinear static behavior can be approximated with TDF by selecting a rational sampling rate.

While these extensions are effective for high level modeling, SystemC AMS has laid out the ground work for reaching out to user defined extensions [4] and even commercial solvers through its synchronization layer [5]. This again is no surprise as the architects of SystemC AMS intended it to be an agile and open source simulator that at times may have to integrate dedicated solvers (Figure 1) for advanced simulation capabilities. The good news is most top of the line EDA tools offer *C interface* that can act as a bridge for cosimulation with SystemC AMS. This work shows how HDL, HDL-AMS variants and SPICE models can be coupled in overall timed data flow simulation under SystemC AMS executable specification using the C level procedural interfaces (VPI and VHPI).

The remainder of this document is organized as follows: We begin with SystemC AMS capabilities in section 2. In the next two sections we compare the merits of system level languages, HDLs, AMS extensions and solvers. We then outline our methodology in section 5 for mixed AMS modeling with SystemC AMS as top level simulator. Section 6 demonstrates an example using mixed AMS languages and abstraction connected to SystemC AMS by the C interface. Section 7 summarizes and concludes our work.

2. SYSTEMC AMS AS AN EXECUTABLE SPECIFICATION

An original promise of SystemC AMS is executable specification. However, as soon as high level architecture is specified, the task to immediately move toward refinement and implementation follows. The easiest way to begin exploring design space and further extract requirements (e.g. for software co-design, analog/digital behavioral partitioning) is to add features in specification or replace abstract/ideal views with more realistic representations. In the experimentation process the digital blocks can be easily cosimulated with RTL or behavioral HDL models. Mixed signal models however stage a greater challenge because of nonuniform sampling [7], nonlinear behavior, mixed domain synchronization and lack of access in EDA tools to connect from outside to the

underlying electrical objects of the model being simulated.

2.1 Synchronization with Timed Data Flow

The TDF is the most sophisticated MoC in SystemC AMS as is it untimed but supports multirate data flow tokens which are equally and discretely time spaced while the solver is executed at each firing of TDF cluster. The firing vector is related to the scheduling of data flow nodes at elaboration.

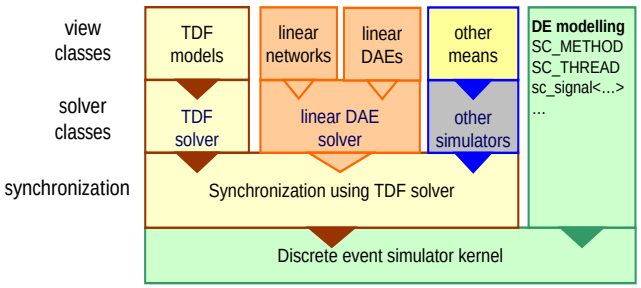

Figure 1: Extendability in SystemC AMS through layered approach [11]

The TDF formalism is a better speed vantage over accuracy for fast simulations of signal processing reigned applications, the main target of SystemC AMS. Further, user defined AMS data types can be defined using the template classes for TDF ports for easing module connectivity. The linear networks are built from electrical primitives inherited from `sca_module` class. Once linear blocks are modeled they need to connect to a TDF domain for signal controlled sources using either SystemC AMS `sca_tdf_signal` or SystemC `sc_signal<double>` nets. This connection is necessary as TDF MoC is responsible for overall tokenization of data in the flow and synchronization of the clusters.

2.2 Abstract Modeling with SystemC AMS

A simple example describing abstract modeling with SystemC AMS is illustrated below:

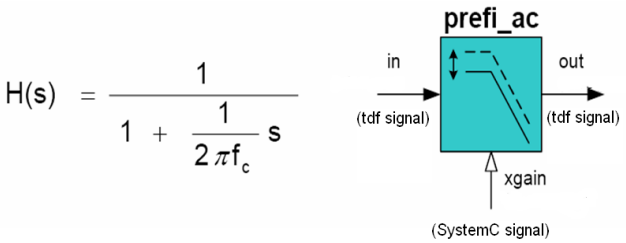

Figure 2: A low pass filter

```
SCA_TDF_MODULE(prefi_ac)
{
  sca_tdf_in<double> in;
  sca_tdf_out<double> out;
  sca_sctdf_in<bool> xgain;
  // parameters
  double prefi_fc; //cut-off freq
  double prefi_g0; //gain !xgain
  double prefi_g1; //gain xgain
  // filter model
  sca_ltf_nd ltf_1; //filter inst
  sca_vector<double> A, B; //coeffs
  sca_vector<double> S;    //states

  void init() {
    //filter coeffs
    B(0) = 1.0; A(0) = 1.0;
    A(1) = 1.0/(2.0*M_PI*prefi_fc);}

  void processing() {
    double tmp=ltf_1(B,A,S,in.read());
    if (xgain.read())
         out.write(tmp * prefi_g1);
    else out.write(tmp * prefi_g0);}

  SCA_CTOR(prefi_ac) { // defaults
    prefi_fc = 1.0e6; prefi_g0 = 2.74; prefi_g1 = 2.74 * 2.2;}
};
```

3. MODELING WITH MIXED ABSTRACTION AND MIXED LANGUAGES

SystemC AMS extensions for analog modeling are sufficient for high level functional models that are abstract e.g. an ideal ADC which represents more of a software view than an AMS one. However more accurate models typically require HDL representation close to implementation. This representation could be at three levels. A higher level of discrete time real valued models of algorithmic nature that can adequately describe mixed behavior digitally with VHDL and Verilog e.g. cycle accurate real valued ADC describing implementation methods with i.e. pipelining, SAR, Sigma-Delta or direct conversion (ADS7818) using comparators. This level of modeling can delineate ADC resolution, dynamic range, throughput, CMRR and quantization accuracy. A second and lower level is of pure AMS behavioral models that characterize physical properties e.g. nonlinearities, timing (acquisition, conversion, response, settling, aperture delay) and numerical errors (gain, SNR, dither, aperture, temperature offset drift). Understanding these behaviors is vital for hardware-software co-designers who need accommodation for tolerances in their designs. However these properties can be extracted with VHDL-AMS or Verilog-A/MS which support structures of analog components and continuous time simulation. These simulators have solvers for conservation laws and differential algebraic equations. The third level is circuit level models described by SPICE that also demand specialty solvers. This is the most accurate and hence slowest level and may not give a respectable advantage at system level over pure AMS models.

Depending the depth of model complexity needed and reflecting its behavior to the top level system specification, the architects may employ any level of HDL/HDL-AMS model granularity for cosimulation with SystemC AMS.

4. ACCURACY, SPEED IN SYSTEMC AMS AND HDL-AMS

SystemC AMS simulation kernel advances in fixed discrete step widths and the solver solves the homogeneous system of equations for the new state estimated in the lapse of time step. Since the systems described are linear approximate, the integration over the step width is reasonably accurate for abstract level and fast simulation. The integration algorithms [8] are likely to be lower order implicit methods to achieve numerical stability (i.e. avoiding extremely small time steps required in stiff explicit methods) but computationally inexpensive. The algorithms cannot be *acausal*

either for using future state estimations e.g. fourth order Runge-Kutta method [10] that is iterative, accurate but not suitable for reactive systems such as real time or embedded e.g. HIL simulation [2].

The alternative is offered by commercial mixed signal tools for nonlinear system solutions with variable step size integration algorithms to accommodate for a range of time constants. These algorithms are more precise and computationally slower in small time range. The stepping is continuously adjusted during simulation depending on dynamic activity zone [10] e.g. large steps during infrequent state changes and small steps to capture rapid state changes. The activity is related to the block requirement (long for transient analysis, small for high frequency poles) [1]. Further the simulation accuracy is covered using various modes e.g. tight conservative, default moderate, relaxed liberal in Cadence Virtuoso Multi-Mode simulator. Similar accuracy based circuit partitioning is implemented in Mentor Graphics' Eldo which also supports large stability range ignoring small variations.

Since SystemC AMS solver is fixed step solver, nonlinearities and nonidealities are difficult to capture. It therefore makes sense to cosimulate such behavior in commercial AMS or SPICE simulator and then relate the results in the top level SystemC AMS TDF view which controls overall simulation synchronization and execution of the commercial simulator.

5. METHODOLOGY CONCEPT

Figure 3 depicts a layered concept of SystemC-AMS and HDL-AMS cosimulation with Cadence tools. The C/C++ based development includes wrappers, OS calls (`fork-exec`) under C/UNIX and TCP/IP socket programming while C level access of HDL objects is possible using the simulator's procedural interface. Further, the open architecture of SystemC-AMS facilitates signal/data type conversions between the two simulated models.

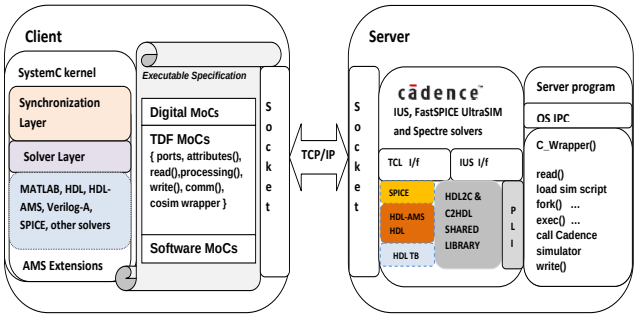

Figure 3: SystemC AMS and mixed HDL-AMS cosimulation interfaces

We have already seen discrete event VHDL [14] and Verilog [16] models can be integrated into simulations using VHPI and VPI respectively. Same methodology can be extended for VHDL-AMS and Verilog-AMS with a caveat. VHPI and VPI were not intended for mixed signal simulation and thus do not adequately address AMS constructs e.g. writing/reading VHDL-AMS `terminal` or Verilog-AMS `electrical` node is not supported. The objects registered with discrete event kernel that can take on real values sampled at discrete times and strictly belonging to digital kernel stream (HDL) can be accessed while objects that vary values continuously with time cannot be directly accessed as per the procedural interface specifications e.g. Verilog-AMS `wreal` although a real valued discrete time wire type cannot be accessed as it is not a Verilog type. Therefore the designer needs to declare the objects (ports or signals) that would be interfaced with SystemC AMS models during cosimulation as HDL types supported by the procedural interface. This is not a limitation in principle as low level signals may not directly connect to system level blocks. The designer however would be interested in their effect on analog behavior and on signal flow to the other blocks.

A cosimulation interface can be invoked in SystemC AMS model within TDF MoC (client). This interface would communicate data between SystemC AMS specified model and the external simulator. The synchronization and firing of the model and consequently of the wrapper i.e. execution of cosimulator is automatically taken care by virtue of the TDF semantics and its connection to the synchronization layer. Another interface as a C wrapper is hosted at the Cadence machine (server). This interface handles C/UNIX system calls for spawning various processes that execute the simulator tool chain using `fork-exec`. It is here where various digital or analog solvers will be called as shown in Figure 4.

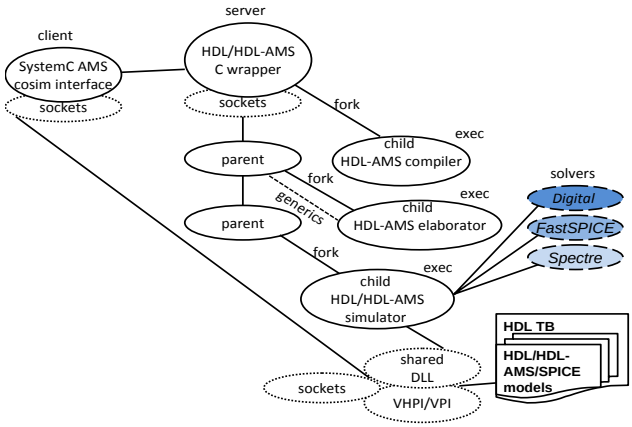

Figure 4: Execution of analog and digital solvers in cosimulation

6. APPLICATION EXAMPLE STUDY

A high level SystemC-AMS description of RF front-end derived from [3] is shown Figure 5. This model is a basic transceiver that consists of a mixer, a low pass filter and a binary amplitude shift keying (BASK) demodulator. The model is a patched with HDL based ADC and then with DAC models. Although SystemC AMS itself can be used to define a behavioral ADC, but we need to go beyond the abstraction level offered by SystemC AMS. Therefore to demonstrate the cosimulation scheme using various solvers we start with a simple real valued HDL model and then successively replace with finer models based on HDL-AMS and even SPICE subcircuits.

A cosimulation interface instantiated in a SystemC AMS model block could be:

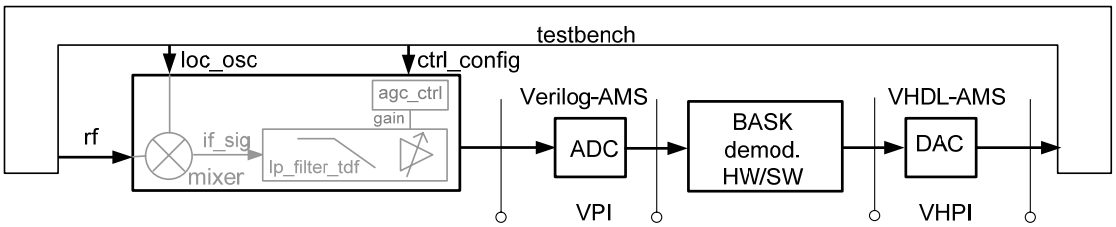

Figure 5: BASK transceiver in mixed SystemC AMS and HDL-AMS descriptions

```
SCA_TDF_MODULE(ad_converter)
{
sca_tdf::sca_in<double> in_tdf;
sca_tdf::sca_out<sc_int<12> out_de;
..
char *out_token_stream;
void processing() {
token_collection = format_and_queue(in_tdf.read());
out_token_stream = cadence_cosim(token_collection);
..
out_de.write<static_cast<sc_int<12> > format(out_token_stream);}
}
```

While execution of AMS/SPICE models at server is:

```
child_pid_ncelab = vfork();
execv("ncelab", "-amsfastspice" "-propspath" "prop.cfg"
"bench_a2d_12bit" "-snapshot" "worklib.bench_a2d_12bit");
child_pid_ncsim = vfork();
execv("ncsim" "-input" "@tcl_script" "-status" "-analogcontrol"
"acf.scs" "worklib.bench_a2d_12bit:behav" "+loadvhpi"
"VHDL2C_DLL" "inst=:bench_a2d_12bit" "+start=0" "+stop=62");
```

The shared dynamic library `VHDL2C_DLL` is the procedural interface application. It applies simulation input generated at SystemC AMS to the VHDL models and also reads VHDL outputs as C data. In addition the library initializes VHDL simulation, registers callback routines, traverses design hierarchy, monitors ports/signals, removes or adds callbacks and synchronizes itself to the VHDL simulation. The library can apply stimulus vectors at the specified times from a file.

6.1 Modeling Levels

Starting from top-down the following model granularities are used in reverse abstraction.

6.1.1 Behavioral HDL Model

The model allows staying in the digital flow because data is real valued sampled at discrete time enforced by the update-evaluate schedule of digital simulator. The solver, a discrete event kernel, based on the event sequence in the schedule solves boolean equations which embed in well defined signal flow [6], therefore there is no analog convergence issue. The refinement flow is still top-down approach. This is the fastest level of cosimulation. However, the model is digital functional oriented one and close to ideal approximation, no critical behavior is captured. Pure RTL blocks or HDL represented analog models requiring clock can be cosimulated at this level. The model can be nicely used for connectivity check or BigD/smallA verification which is traced back to the cosimulation with SystemC AMS executable specification. A real valued behavioral VHDL of 12-bit ADC (ADS7818) is cosimulated under TDF. VHDL outputs are read using VHPI.

6.1.2 HDL-AMS Model

VHDL model is replaced with VHDL-AMS and Verilog-AMS models. The solution is furnished by the analog kernel (FastSPICE or Spectre) while VHDL-AMS real valued discrete time outputs are read using VHPI and Verilog-AMS real valued discrete time outputs are read using VPI. The HDL-AMS models are based on ODEs in time and frequency domains which are more accurate than pure HDLs and are an entry point for *meet in the middle* approach. Verilog-AMS is better suited for mixed signal simulations due to being weak type language. The simulator automatically resolves implicit disciplines by inserting interface elements between signal domains of analog and digital models [9]. These elements can be explicitly insert by the user to aid in resolution. However for VHDL-AMS the conversion models must be written [15].

6.1.3 Mixed HDL-AMS Models

Next a mixed HDL block comprising of 12-bit ADC and DAC pair is simulated. Using two C wrappers SystemC AMS is interfaced with Verilog-AMS ADC (VPI) and with VHDL-AMS DAC (VHPI). For a unified mixed HDL simulation Cadence irun tool is used.

6.1.4 SPICE in the Middle

To better study accuracies and nonlinearities a SPICE subcircuit or Verilog-A submodel can be inserted in the overall cosimulated model. Parts of RF front-ends that serve as boundaries between digital baseband and high frequency channel can be modeled with active devices. The migration toward implementation can become more visible at such abstraction level since vendor specific transistor level SPICE definitions and parasitic device models can be used. Such a model can be a pure SPICE netlist or a mix of SPICE and HDL-AMS model. However, the model must be instantiated at a low hierarchy in a HDL/HDL-AMS block for SystemC AMS interfacing using C interface (VPI/VHPI) and for digital stimulus. For example, the HDL could be a testbench. The model is the better approximation to the real world. This is the slowest and most accurate level and a turning point for upcoming bottom-up analog design to be handcrafted due to absence of analog synthesis. It should be noted that SPICE modeling is not a replacement of full-scale RF simulations that require precise transistor level models and use fine stepping for understanding heterodyne receiver sensitivities and interference coupling. SPICE models only

represent reduced order of nonlinear macromodels [12] to give orders of magnitude in simulation speed. Such reduction has to be handmade if there exists no methodology or tool for a direct extraction.

In SPICE in the Middle arrangement a Verilog-AMS block instantiates a SPICE block that, in turn, can instantiate a Verilog-AMS block. Hence a Verilog wrapper is typically required when a VHDL testbench instantiates a SPICE netlist. A port mapping file binds Verilog and SPICE ports. The analog simulation is controlled by analog *controlfile* passed to the HDL-AMS (NCSIM) or irun utility for integrated one-step simulation.

6.2 Analog Solvers

The computational capabilities are stepped-up by solving the equations derived from HDL-AMS and SPICE descriptions using the following Cadence solvers. Note that the overall effect of the solved/cosimulated model is projected over the SystemC AMS abstract description.

6.2.1 Spectre

The Spectre solver is an enhanced SPICE simulator using similar solution methods as in traditional SPICE (Newton-Raphson, implicit integration and sparse direct matrix) but improved, faster and accurate versions. SpectreRF allows simulations of nonlinearities in analog and RF mixers, S/H, oscillators designs. The high carrier frequency models e.g. RF modulators and demodulators in which transient analysis is a simulation bottleneck for solving at fine step size, *envelope* analysis is a better mode. In this mode digital blocks are dealt with digital solver and analog blocks are simulated using available envelopes e.g. oscillator, FM, Newton. The two simulations are always synchronized at the next analog simulation point.

6.2.2 FastSPICE

The FastSPICE gives near SPICE accuracy for simulating large scale circuits. The FastSPICE (UltraSim) solver simulates mix baseband, IF and RF frequency systems. The time step resolution is determined by the highest frequencies present which essentially makes too many time steps and unnecessary overhead for low frequency signals. FastSPICE uses transient analysis with envelop designation for high frequency circuit areas. Predesigned blocks can be cosimulated with FastSPICE to import critical behavior in SystemC AMS.

7. CONCLUSION AND FUTURE WORK

A gentle methodology was presented that substantially empowers SystemC AMS abstract description by incorporating simulation results computed by strong commercial solvers. The typical problem of synchronization associated with simulator coupling is engrossed by the TDF MoC of SystemC-AMS master simulator which is responsible for firing dataflow nodes and thus automatically invokes the cosimulation task. The access to cosimulated data is delivered through the procedural interface which are available in all popular simulators. The dedicated simulator/solver enables modeling of implementation details and nonlinearities that normally cannot be gained by a system level simulator e.g. SystemC and/or SystemC AMS.

Future work would carry on implementation details, simulation results and discussions.

8. REFERENCES

[1] Accelerating Analog Simulation with Full SPICE Accuracy, White paper, Cadence Design Systems, Inc., September 2008. Available online (7 pages), `http://www.cadence.com/rl/Resources/white_papers/spectre_turbo_wp.pdf`.

[2] C. G. Alain Vachoux and K. Einwich. SystemC-AMS Requirements, Design Objectives and Rationale. *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 388–393, 2003.

[3] A. V. Christoph Grimm, Martin Barnasconi and K. Einwich. An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS Extensions, June 2008. Available online (12 pages), `http://www.systemc.org/downloads/drafts_review/`.

[4] C. G. M. D. F. Herrera, E. Villar and J. Haase. Heterogeneous Specification with HetSC and SystemC-AMS: Widening the Support of MoCs in SystemC. In *Embedded Systems Specification and Design Languages: Selected Contributions from FDL'07*, pages 107–121. Springer, 2008.

[5] C. Grimm. Modeling and Refinement of Mixed-Signal Systems with SystemC. In *SystemC: Methodologies and Applications*, pages 299–323. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[6] W. Hartong and S. Cranston. Real Valued Modeling for Mixed Signal Simulation, App Note, Cadence Design Systems, Inc., January 2009. Available online (20 pages), `www.cadence.com/rl/Resources/application_notes/real_number_appNote.pdf`.

[7] S. Joeres, H.-W. Groh, and S. Heinen. Event Driven Analog Modeling of RF Frontends. pages 46–51, September 2007.

[8] C. G. Karsten Einwich, Peter Schwarz and C. Meise. SystemC-AMS: Rationales, State of the Art, and Examples. In *SystemC: Methodologies and Applications*, pages 273–297. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[9] K. Kunder and O. Zinke. *The Designer's Guide to Verilog-AMS*. The Designer's Guide Book Series. Springer, Berlin, 2004.

[10] J. Ledin. *Simulation Engineering:* Build *Better* Embedded Systems *Faster*. R&D Developer Series. CMP Books, Lawrence, KS, USA, 2001.

[11] C. G. F. H. Markus Damm, Jan Haase and E. Villar. Bridging MoCs in SystemC Specifications of Heterogeneous Systems. *EURASIP Journal on Embedded Systems*, Vol. 2008(Article 738136):16, May 2008.

[12] J. Roychowdhury. Reduced-Order Modeling of Time-Varying Systems. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 46(10):1273–1288, October 1999.

[13] W. H. A. Schilders and E. J. W. ter Maten, editors. *Special Volume: Numerical Methods in Electromagnetics*, volume 13 of *Handbook of Numerical Analysis*. 2005.

[14] J. Shields. Modeling Foreign Architectures with VHPI. In *VIUF '00: Proceedings of the VHDL International*

Users Forum Fall Workshop (VIUF'00), pages 100–107, Washington, DC, USA, 2000. IEEE Computer Society.

[15] J. Shields and E. Christen. Mixed Nets, Conversion Models, and VHDL-AMS. In *Advances in Design and Specification Languages for SoCs: Selected Contributions from FDL'04*, pages 100–112. Springer, 2004.

[16] S. Sutherland. *The Verilog PLI Handbook: A User's Guide and Comprehensive Reference on the Verilog Programming Language Interface*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.