Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2010, Article ID 489365, 13 pages
doi:10.1155/2010/489365

Research Article

On Mixed Abstraction, Languages, and Simulation Approach to Refinement with SystemC AMS

Yaseen Zaidi, Christoph Grimm, and Jan Haase

Institute of Computer Technology, Vienna University of Technology, Gusshausstraße 27-29/E384, 1040 Vienna, Austria

Correspondence should be addressed to Yaseen Zaidi, zaidi@ict.tuwien.ac.at

Received 31 May 2009; Revised 21 September 2009; Accepted 31 December 2009

Academic Editor: Sergio Saponara

Executable specifications and simulations are cornerstone to system design flows. Complex-mixed-signal embedded systems can be specified with SystemC AMS which supports abstraction and extensible models of computation. The language contains semantics for module connections and synchronization required in analog and digital interaction. Through the synchronization layer, user defined models of computation, solvers and simulators can be unified in the SystemC AMS simulator for achieving low-level abstraction and model refinement. These improvements assist in amplifying model aspects and their contribution to the overall system behavior. This work presents cosimulating refined models with timed data flow paradigm of SystemC AMS. The methodology uses C-based interaction between simulators. An RTL model of data encryption standard is demonstrated as an example. The methodology is flexible and can be applied in early design decision tradeoff, architecture experimentation, and particularly for model refinement and critical behavior analysis.

1. Introduction

The exploration of design space with model refinement by reusing legacy models requires developing an executable specification before a frozen *product shall* specification. Analyzing alternate system configuration and rearranging of model blocks depend on putting together quick virtual prototypes of the system. Immediate try out of various models in the executable specification is possible with high simulation speed and relative ease of model plug-in. Decomposing models into detail implementation options exposes problem areas, helps resolve conflicting requirements, and causes greater visibility in design by weighing design feasibility with goals such as critical analysis (corner case coverage, worst case analysis, failure analysis, and fault tolerance), optimization (physical area, power consumption, or speed) and performance evaluation (speed, latency, throughput, bandwidth, response time, link utilization, BER, and QoS).

Our work presents an approach to replace abstract model blocks described in SystemC AMS with fine grain models written in a hardware description languages. We cosimulate the two domains of Models of Computation (MoC). The methodology is pure simulation based and does not rely on controlled stepwise refinement or formalism. The goal is to directly and effortlessly insert specific models in the system and quickly generate architectures that can be traded off on merits of specification, correctness, performance, and accuracy to yield an implementable platform. This type of refinement needs a model rich library of possible implementation choices. SystemC AMS is an executable specification-based analog mixed signal simulator that supports varying abstraction levels. We present, using scalable cosimulation, a test case of abstract SystemC AMS description, in which a parity check block is refined as a synthesizable VHDL model of Data Encryption Standard (DES). The description is Amplitude Shift Keying (ASK) transceiver which originates from the ANDRES project [1] whose goal is to reduce the design time and cost of highly integrated embedded systems.

This work covers certain novel aspects. Until now previous works have targeted refinement in pure digital environment; however we do direct digital refinement in heterogeneous/mixed signal systems. In the simulation domain, the older works use VPI C interface [2, 3] but to our knowledge we do not know of any published work based on the VHPI interface. SystemC AMS does not require commercial licensing of the costly system design tools (Section 3) to the technical community. SystemC AMS is much faster than the conventional system design tools. Further more the cosimulation interfaces in commercial tools are proprietary and inadequate in most academic tools; whereas SystemC AMS tackles these issues by being open source, C/C++ based, and through its synchronization layer.

2. Related Work and Methodologies

Methodologies for refinement as it relates to design space and architecture exploration use several methods which can be categorized as formal and informal. For example for more controlled and formalized refinement, graph-based formalism (graphs of problem, specification, architecture, and network) is at the corner stone of SysteMoC [4]. These graphs are used for binding and mapping in feasible implementation, optimization, and resource allocation during refinement. A similar symbolic representation is in [5] which reduces the exploration issue to a linear programming optimization problem using pseudoboolean solver. Orthogonality [6] driven refinement (behavior, timing, and interface) has also been proposed. In an earlier technique [7] the refinement task is split into types: control, data, and architecture. Dataflow graphs and trace transformation of events were utilized in [8]. Among the less formal methods, a holistic way of interconnecting modules [9] uses UML SysML. Interface-based [10] partitioning of networked architecture is suited for NoC applications. Automatic IP selection was proposed by [11]. A stepwise exploration flow (steps of analysis, building, exploration, composition, estimation) is discussed in [12]. A framework using hardware emulation [13] has also been suggested. Simulation-based methods are widely spread [6, 14, 15]. Many works take MPEG [4, 13, 16] and JPEG [8, 17, 18] as test cases. In compliance to the SystemC AMS philosophy of abstraction our work uses simple cosimulation-based refinement embedded in Timed Data Flow (TDF) graph while not requiring complex mechanics of interfacing and therefore the technique is informal.

3. Related Tools

Architecture development, design space exploration, and model refinement all are generally tool driven processes. These concepts are closely related and thus there is significant thrust from the technical community to research them. The effort can be broadly categorized in industrial and academic realms each pushing with their own tools and neither providing a complete solution because of the shear size of the problem.

In the industry architecture design, refinement and partitioning is supported using automation by Electronic System-level (ESL) tools which predominately target digital domain of the heterogeneity because it makes the largest contribution in design space. The ESL tools are: CoWare Platform Architecture, ARM RealView, Bluespec Development Workstation from Bluespec, Arteris NoCExplorer, Mirabilis VisualSim, Tensilica XPRES Compiler, AutoESL AutoPilot, Binachip-FPGA tool, Mitrion Platform, and Synfora PICO Express FPGA among others. Most of these tools support hardware software codesign using either virtual platforms or FPGA prototypes. The problem of refinement is usually labeled as high level synthesis which basically means to generate a synthesizable circuit description from algorithmic or behavioral description, that is, a translation process for converting untimed SystemC-based description into equivalent HDL description. The translation also converts cycle accurate models of datapaths, control units, memory banks, busses, and interfaces to bit accurate models. The HDL description can then be input to other refinement engines and applied with architectural design constraints to produce even lower levels of abstraction for example RTL, or gate level netlist mapped to a target technology. High level synthesis is supported by Cadence C-to-Silicon Compiler, Mentor Graphics CatapultC and Seamless, Synopsys CoCentric, SystemCrafter SC from SystemCrafter, and Cynthesizer from Forte Design System. These tools are a big jump forward to bridge the gap between macroarchitecture and underlying mapped microarchitecture since SystemC is suited for design exploration. The only drawback is that the HDL description is not generic or clean enough to readily target any vendor's FPGA fabric or ASIC standard cells. High level synthesis tools and ESL tools are coupled by custom signal processing tools, for example, MATLAB/Simulink, Cadence Signal Processing Worksystem, or Virtuoso. The motivation has been prevailing wireless-based products. Instruction Set Simulators are used for running virtual firmware on processor core. The industrial based designs are centered around IP cores of Multiprocessor System on Chip (MPSoC) and on-chip bus communication. The bus connects processors, power management unit, timing units, memory blocks, microcontrollers, DSPs, external busses, for example, USB, Ethernet, and supporting peripherals. The processors' architectures are generally RISC based, for example, ARM, PowerPC and bus architectures are AMBA or NoC. SystemC interface classes and Transaction Level Modeling (TLM) are a major modeling paradigm for bus topologies.

The academic research also employs SystemC-based tools. Academic circles use established tools such as Metropolis, Ptolemy and its variants, for example, Kepler for signal processing applications, mathematical intensive models and MoC calculations. LESTER-UBS aims at reconfigurable architectures while MILAN framework is integrated with MATLAB, HiPerE and DESERT (formal methods). SESAME simulator uses Kahn Process Networks (KPNs) for concurrency and Y-Chart for abstraction level and domain recognition. High level systems can be described by LISA language which is an architectural description language. SystemCoDesigner takes specification graphs as inputs and generates platform-based virtual prototypes covering specification, automatic exploration; and implementation. The academic groups try to advance research while commercial tools manufacturers try to earn revenues on their R&D work, however the marriage of academic and commercial tools is a rarity and the gulf is large. Some exceptions are that UC Berkeley Ptolemy dataflow modeling paradigm was incorporated in Agilent ADS simulator, Ptolemy from Agile Design has found wide acceptance in many large companies, Bluespec tools are written in Haskell, and SystemC was influenced by SpecC.

A major drawback in the tools is the lack of analog modeling support below system-level.

4. Modeling with SystemC AMS

4.1. The Right Level of Abstraction. The motivation behind analog and mixed signal extension to SystemC has been the modeling of applications that are dominated by signal processing models [20], better understanding hw/sw interaction and developing concepts of AMS systems at architectural level [21]. With this in mind SystemC and SystemC AMS opted for C/C++ based classes and methods for their kernels only augmented with hardware data types and concurrency. SystemC is now well settled in the EDA community because of the immense advantage it brings over the conventional HDL simulators inadequacy as hardware faithfully follows Moore's law and is more and more interlocked with software.

SystemC AMS-based modeling too has been on rise. Design techniques with SystemC AMS have been presented by [22]. Modeling of wired communication system [23] and wireless nodes [24] have been reported. I^{2}C bus communication [25], electromechanical [26], acceleration sensor [27], inertial navigational [28], and nonlinear dynamic systems [29] have been simulated with SystemC AMS.

The current SystemC AMS prototype offers three MoCs as shown in Figure 1: 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 at appropriate time. Simple nonlinear static behavior can be approximated with TDF by selecting a rational sampling rate. External solvers [30] can be interfaced with SystemC AMS as well as user defined MoCs [31] can be formally binded. Analog TDF models can interact with TLM-based DE models [32]. These capabilities are due to the open architecture of SystemC AMS. Figure 2 shows a typical SystemC AMS cluster which is a set of binded modules/nodes solved by same MoC such as TDF.

4.2. Timed Data Flow in SystemC AMS. SystemC AMS is a class library and analog mixed-signal extension built as an add-on to SystemC digital simulator. SystemC AMS supports high level model abstraction with Time Data Flow (TDF) MoC which perfectly suits for loose simulator coupling. TDF is an analog solver that abstracts away cycle accurate timing information from parent SystemC, a Discrete Event (DE) simulator. TDF tokens are equally time spaced whose sampling rates are determined by the system clock [33], that is, constant analog stepping width (number of clock cycles) is mapped to SystemC `sc_time()` class. This solver does not support adaptive or variable stepping yet in the current prototype as in commercial analog simulators that trade accuracy with speed. At system-level an analog simulator is too slow because of the integration time problem. These solvers would compute nonlinear system of equations for the model or netlist at each integration point. The simulation would be even slower for large circuits. Therefore for system-level AMS designs it is best to use a DE driven simulator (SystemC) with integration-based simulator (SystemC AMS). This combination enables software codesign as well which is not possible with SystemVerilog. The constant size stepping gives further simulation efficiency because the computational overhead of determining new step size is absent. TDF scheduling is static (ABCCCCCCCCC in Figure 2) for determining execution order of MoCs in the data flow network. This order is set at cluster (connected modules with common MoC) elaboration phase before simulation beginning. A static schedule or firing vector is calculated once and is applied repeatedly in every firing.

4.3. Synchronization Layer. The synchronization of analog TDF models to DE models as well of any user written special solvers or externally hooked foreign simulators is AMS synchronization layer [34] duty which registers all DE and TDF MoCs and their data (signals, variables). The synchronization layer shown in Figure 1 determines execution order of the solvers and simulators in fixed time steps and synchronizes DE and TDF blocks in these steps. The analog and digital models are individually solved and synchronization of analog and digital domains is resolved by relaxation method [35] with little overhead since interactions are discrete events.

5. Refinement Using Cosimulation

5.1. ASK Transceiver—A Case Study. The ASK-based systems use On Off Keying (OOK) modulation for conserving power during off cycles. This scheme is simple and inexpensive. These systems are implemented in remote keyless entry, tire pressure monitor, and antilock brake systems of the automobiles.

The ASK transceiver shown in Figure 3 is a nonconservative high-level analog mixed signal model. Its subsystems are modeled in TDF and DE domains. The analog TDF blocks are shown with the shadows. The waveform envelopes and silences are detected as bits. The task is to replace abstract cryptographic unit with a legitimate and realizable model. The selected models are VHDL RTL descriptions of DES and Triple DES (3DES) algorithm. The DES algorithm, released by the National Institute of Standards and Technology (NIST), encrypts message space M into the code space C using a unique key K, where the message, code, and key words are all 64 bits. Let e and d be the encoding and decoding functions, then the permutation cipher is one of the $2^{64}!$ permutations on M. The encoding and decoding functions are

$$\begin{gathered} e \in K \longrightarrow M \longrightarrow C, \\ d \in K \longrightarrow C \longrightarrow M, \end{gathered} \tag{1}$$

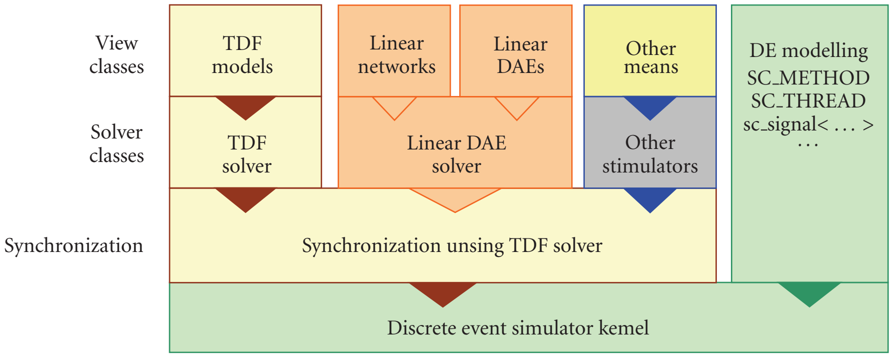


FIGURE 1: Extensibility in SystemC AMS through layered approach [19].

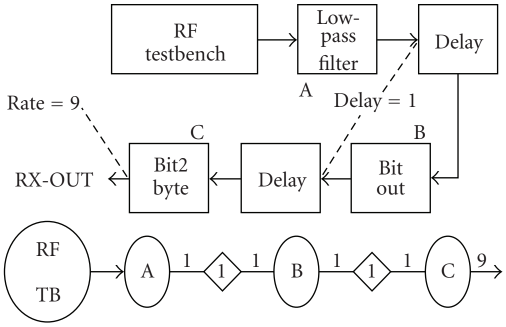


FIGURE 2: A cluster of TDF nodes and firing vector ABCCCCCCCCC.

where $K = 2^{64}$. K is selected as one out of 2^{56} permutations that are most random among 2^{64} total available. The remaining 8 bits may be used as parity bits for error detection.

The key is changed frequently to reduce the risk [36]. In theory the key can be determined by 2^{56} or more than 70 quadrillion (70×10^{15}) operations. The process can be made more robust with 3DES which requires 2^{112} operations to crack the key although noncomputational schemes of hacking, for example, power analysis [37] have been known. Encryption and decryption processes use the same key and algorithm; however the subkeys are generated differently.

Any legacy or refined model must be golden. The DES/3DES RTL model is first verified stand-alone in loop-back mode with several NIST test vectors. The model is further verified with a logic equivalence checker that compares RTL synthesis and physical synthesis netlists. The mode of DES operation is Electronic Codebook (ECB).

5.2. Simulation Interfaces. In the crypto unit module, we replace model description below ports and attribute definition with a cosimulation interface. The interface transforms the tokens and attributes associated with the TDF input ports into strings. The formatted data is then sent to an open socket at a remote computer that houses VHDL tools. The external simulator applies the received inputs to the refined VHDL model (DES) of the crypto unit. When VHDL simulation is finished the outputs are received in the crypto unit block which drives it to its output ports. The cosimulation interface is a blackbox in the module which only provides an interface to the remote simulator, where the refined model will be evaluated. SystemC AMS and VHDL simulators are configured in client-server topology on two computers. Distributed cosimulations are often a necessity because tools consisting of dedicated solvers have best performances on specific operating systems and different machine architectures. Various layers of applications that play role in cosimulation are depicted in Figure 4 which shows interface boundaries in the distributed cosimulation topology. SystemC AMS acts as a master simulator running at the client computer which connects to the VHDL simulator at the server computer. The executable specification, that is, a SystemC AMS model, may contain analog, digital, and software models. On the server side, a daemon runs which makes use of UNIX processes (parent-child) and system calls (`fork-exec()`) to invoke Cadence tools. Each child process, after calling its particular Cadence tool, does a control transfer to another process to call the next tool in the suite. The last child process calls the simulator and after completing the simulation returns the control to the parent process. A shared C library accesses VHDL objects.

The distributed simulation runs in full automation requiring no user intervention. Two binaries are obtained. The main SystemC AMS description is compiled with the client cosimulating wrapper discussed in subsection V.D. Another wrapper runs at the server as a simple program. Both wrappers communicate via TCP/IP sockets. To compute the MoC, the client wrapper passes data tokens and necessary information to the server wrapper which then sets up the simulator after scrutinizing the received information. Setting up the simulator suite is a major task since a series of tools has to be called in the right order and with a list of command options and arguments. This job can be done in two way: wrapper invoking a setup script written in simulator's native scripting language; embedding the setup calls in the wrapper as if they would be made from the simulator command line interface. Besides the data token, the information related to MoC evaluation such as any other signals required in computation that do not originate from the SystemC AMS description, for example, clock, reset is

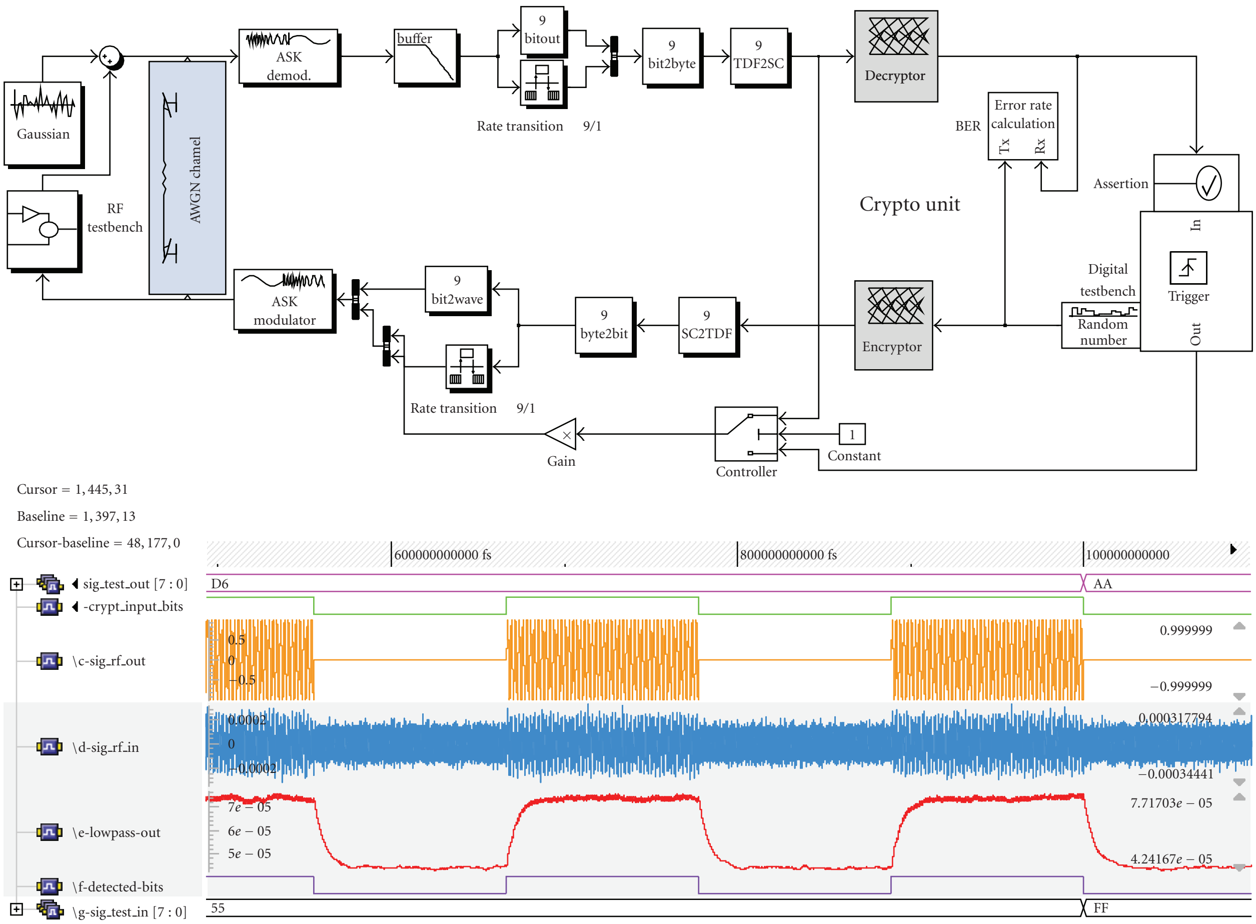


Figure 3: ASK transceiver model and simulation.

generated in the local testbench at the server. The testbench is an important element in cosimulation; it must drive the model with properly timed and locally generated signals. This is because the model to be evaluated on a mature and commercial simulator is finer than the system-level model and thus needs more input signals than simply the input data. Furthermore, the refined models should never be modified for the sake of cosimulation.

A description of the crypto model with cosimulation C wrapper is shown in Algorithm 1.

5.3. Synchronization. The TDF formalism in SystemC AMS is realized using blocking read and write of `sc_fifo()` primitive channel of SystemC in which reading from an empty FIFO or writing to a full FIFO suspends the calling process until data are available to read or FIFO has space for new data. In multirate TDF the FIFO depth is related to the rate attribute of the TDF. The rate value therefore must be realistic and restricted by both solution of homogeneous equations of the TDF cluster for determining a valid firing schedule and machine resources. Thus no tokens are lost in cosimulation.

The clocks of the two kernels run completely independent of each other. Their timing states are not passed to each other. In fact the timing at SystemC AMS is coarse since the tokens however large are sample accurate or cycle approximate (the values are guaranteed at the end of the cycle ignoring timing information during the cycle) sampled at the system clock whereas on the VHDL side every bit is clocked (cycle precise) making the sample bit-accurate in addition to cycle-accurate. Such timing contrast is ideal for system-level simulation where speed is necessary and for refinement where problem boundaries can be pin pointed. Once the inputs have been available the cosimulation interface executes. However, the computation of outputs takes some time at the cosimulator and new tokens remain pending for next simulation iteration. This latency by virtue of the nature of TDF dataflow takes care of time warp problems in simulations: rolling back [38], lazy re-evaluation [39], cancellation [40], runahead (e.g., Calaveras algorithm by Synopsys); and slower lock-step algorithm in selecting smallest step for forced synchronization. The execution order of each TDF block is preset and the graph is cyclically directed; the cosimulator follows the order determined by SystemC AMS static scheduler at elaboration [33]. The reaction time creates a temporary deadlock in the TDF cluster because the consuming actor must wait for the producing actor to fire. The deadlock state is cleared as soon as the tokens arrive from the refined model. The

```
/* SystemC AMS (client) */

SC_MODULE(crypto_unit) {        // crypto unit module
    sc_in<sc_bv<64> >    Rx;    // dig. encoded I/O
    sc_out<sc_bv<64> >  Tx;
    sc_in<sc_bv<8> >  in;       // dig. unencoded I/O
    sc_out<sc_bv<8>>  out;
    sc_in<Mode> cmode_in;       //  control I
    Mode cm;                    // current power mode
  SC_CTOR(crypto_unit) {
    cm = des;                   // default is DES
  SC_METHOD(decode);            // method for decoding
    sensitive << Rx;            // decode when Rx updates
  SC_METHOD(encode);            // method for encoding
    sensitive << in;            // encode when in updates
  SC_METHOD(cryptoControl);     // sets the mode
    sensitive << cmode_in; }
  void encode() {
/* convert bit vector to integer values */
token = in.read().to_uint();
..
/* initiate comm link and cosimulation at server */
rtoken_tx = wrapper_CadenceIUS_TX(&to_str64(token));
{..
status_code = write(sock, msg_token, Len);
..
to_SystemCAMS_type(rtoken_tx);
..
Tx.write(encrypted_val);}
..
void decode()
..}
```

Algorithm 1

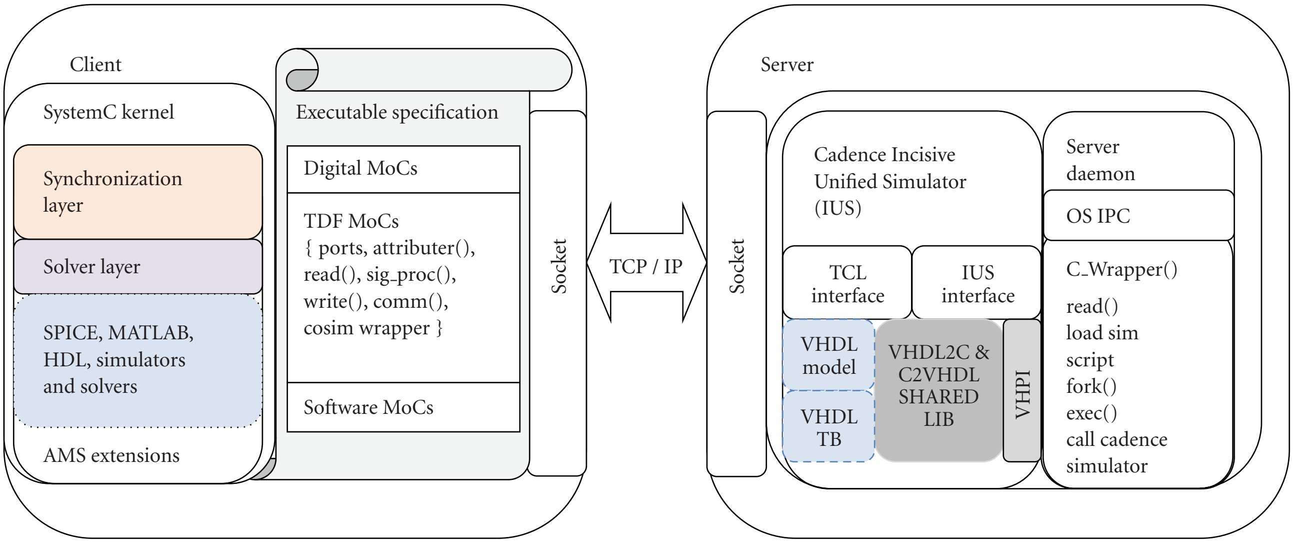


Figure 4: Cosimulation interfaces.

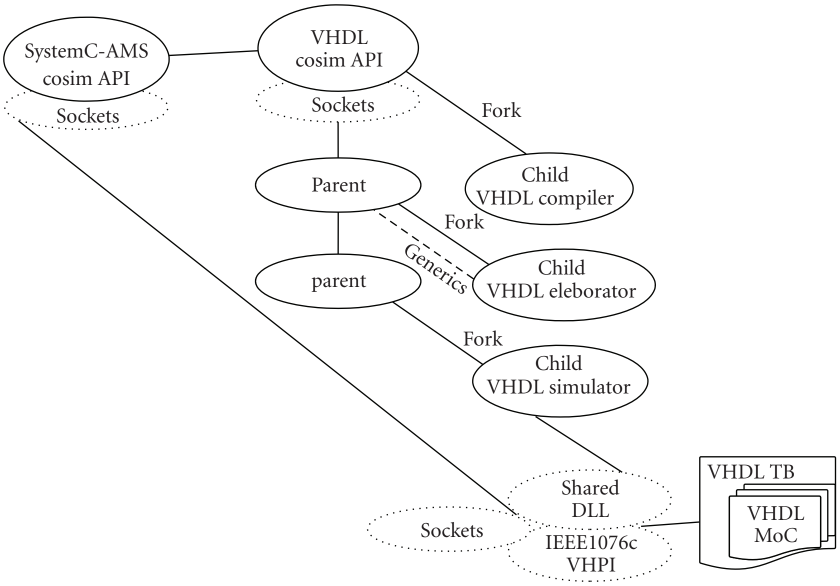


Figure 5: VHDL cosimulation process hierarchy.

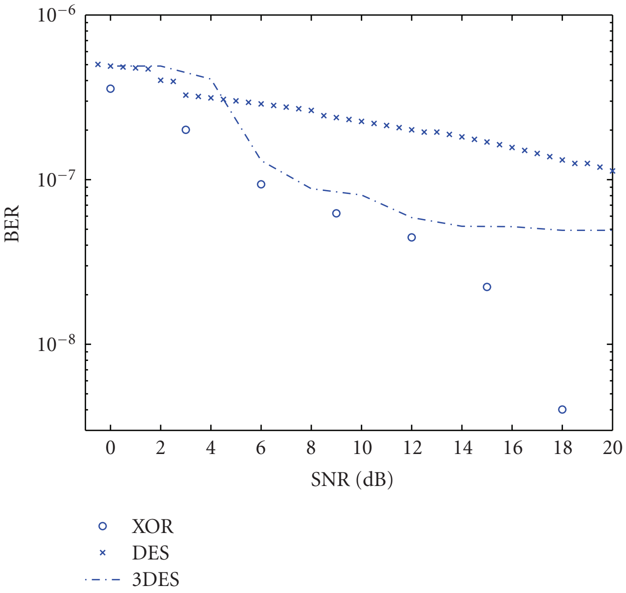


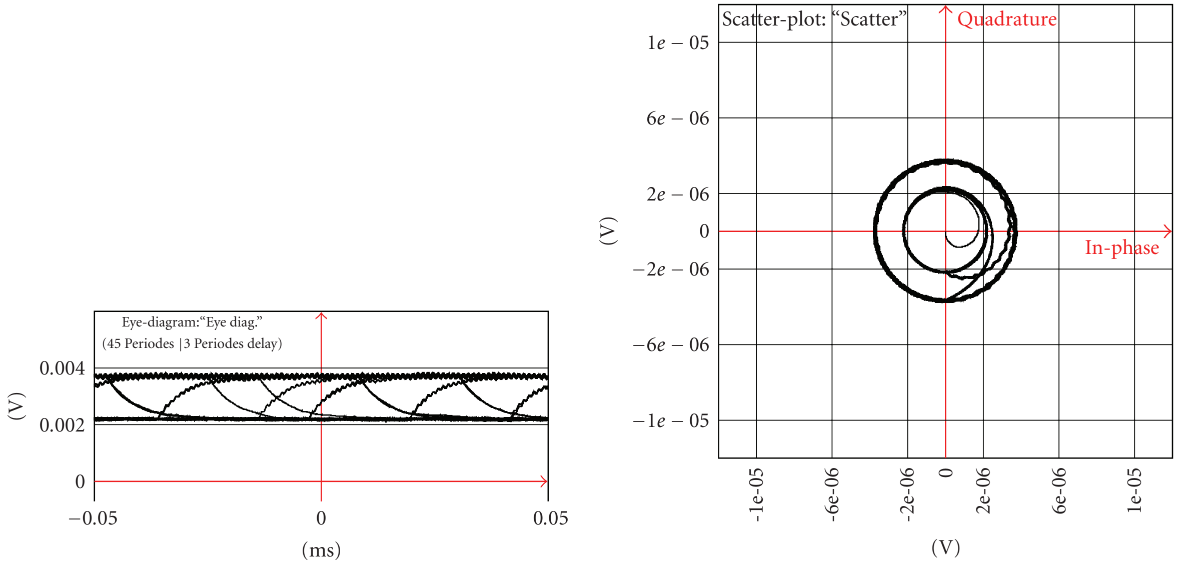
Figure 6: Bit error rate in ASK transceiver using different encoding methods.

deadlock is different than the deadlock caused by cyclic dependency of the balance equations in the TDF cluster system which is precluded by slipping in a finite delay in the cycle [41]. The process is completely asynchronous for the VHDL cosimulator while only locking is at the master SystemC AMS simulator which momentarily suspends itself for the cosimulated actor node while the simulator may be busy solving for nodes that have data. The static scheduling vector is applied for every simulation iteration.

5.4. MoC Computation. The external simulator is handled by a daemon program which communicates with the SystemC AMS master. The daemon validates all incoming tokens and data (length and bit integrity of C-strings) required for refined MoC computation. This information is transformed in semantics for the VHDL language. For example, the port rate in the multirate TDF would be mapped to the number of clock cycles needed to write the output tokens for the corresponding input tokens to maintain the balance in the dataflow graph.

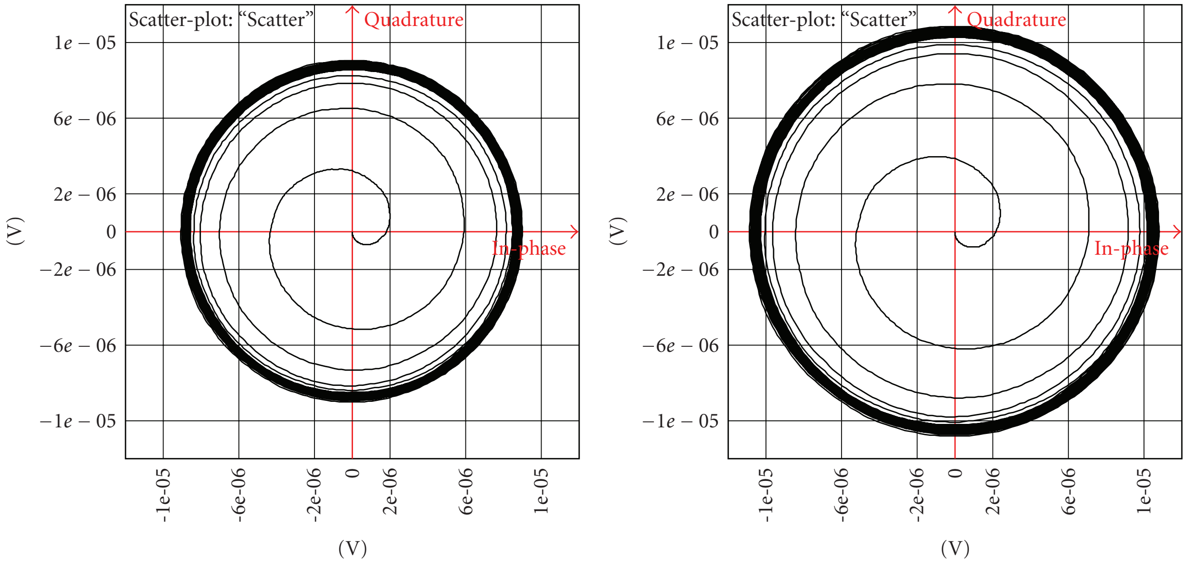
Using OS utilities the daemon runs several child processes each executing a specific tool of the simulator tool flow as shown in Figure 5. Since the model is computed in a static fashion, that is, the input token values do not change throughout the simulation, these values can be passed to the VHDL elaborator as C-strings which will apply them to the model before simulation. If there are more than one input tokens that need to be fed periodically to the model, then they need to be input using the C interface by updating the input ports as discussed in Section 6.

At higher level of abstraction such as the one used for ANDRES usually a clock signal is not needed for modeling since much of the behavior can be modeled reasonably without clock as clock makes simulation slow. However, since abstraction granularity is introduced in cosimulation using HDL, indeed clock becomes inevitable to drive HDL part of simulation. A testbench clocks the VHDL model and converts the values of SystemC AMS semantics of port rate, delay, and so forth, to meaningful VHDL objects if required by the VHDL model, for example, the rate attribute can be used for a counter to pulse an `enable` signal. Additionally the testbench times and drives essential ports in the refined model because they are local to the VHDL design and orthogonal to



(a) (b)

Figure 7: Output of LPF analyzed for noise (9-bit XOR).



(a) (b)

Figure 8: Demodulated signal quality observed at the output of LPF for DES and 3DES in transmission path.

the abstract AMS description. The testbench converts C-strings of inputs into equivalent VHDL standard logic types. The testbench is an interface to bridge the abstraction in semantics of SystemC AMS and VHDL models.

Next 3DES model is simulated. The model requires no change to the interface between the two simulators. However, the VHDL testbench is modified to provide two additional encryption keys so that all three cypher operations in the cascaded stages have their unique keys. An Initialization Vector (IV) is also required in the testbench to start the process because 3DES computation depends on the previous computation. The IV is symmetric for encryption and decryption.

6. C Level Access

Upon receiving messages the daemon reacts and executes simulator at server as follows. The `execv()` calls successively invoke VHDL compiler, elaborator, and simulator. The simulator loads the C library for VHDL module access with `+loadvhpi` argument (see Algorithm 2).

The data tokens are exchanged between the daemon and the VHDL simulator by sharing the simulator's C

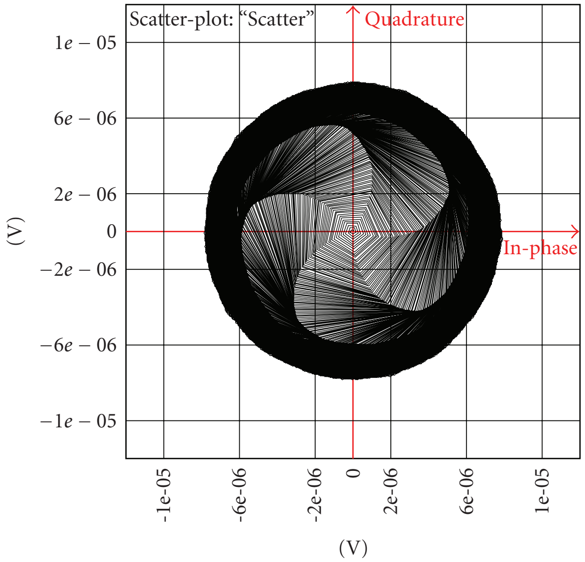


Figure 9: Phase difference at the output of LPF for 3DES case (LO frequency = fc).

interface IEEE Std 1076c-2007 (VHPI). This task is handled by creating an application for access of VHDL objects (instance, hierarchal components, ports, signals, data). The application is a shared library dynamically linked to the simulator executable and loaded by the particular child process of the daemon. The library contains callback functions for initialization, navigating VHDL design hierarchy, assigning ASCII IDs to the ports, registering simulation time, adding/removing value change callbacks, and storing output data and their time stamps. The library also tracks the simulation time and makes access to the VHDL outputs as they change values (VCD dump). Although recording values on transitions are an efficient way, our methodology is flexible to read off values at any specified times or stops in addition to value changes. The library sends the read data to the SystemC AMS simulator connected through socket. All ports are monitored but only the ports of interests are accessed and their outputs saved. The simulation start and stop times needed for the computation are preprogrammed and passed to the simulator by the child process. The designer should know approximately when the outputs are stable, valid, or meaningful to control the simulation.

All data needed for MoC computation in AMS description at client is converted C-strings to be accepted by the socket. Therefore the input data and signals (SystemC or SystemC AMS hardware types) are formatted using overloading, static type casting, and other built-in C/C++ constructs. At the server the C interface reads VHDL ports as C-strings which are converted to SystemC/SystemC AMS types back in the main AMS simulation. Thus native C-based translation between SystemC/SystemC AMS and VHDL rules out special signals [1] and converter channels [19]. The C based VHPI and Verilog VPI interfaces might find greater acceptability in mixed language cosimulations with the advent of SystemC/SystemC AMS (see Algorithm 3).

7. Simulation and Experimental Results

As soon as a token arrives in the crypto unit of SystemC AMS description, the cosimulation interface is invoked to simulate the DES model. The cosimulation interface on SystemC AMS side handles necessary communication and formatting, for example, zero padding for forming a 64-bit word for encryption, stripping off leading zeros of 64-bit decrypted word, making sure all incoming strings characters are 1s or 0s and the size of the received and transmitted data conform to the expected buffer lengths. Both DES and 3DES RTL descriptions are cosimulated. The simulations runs in complete automation. The methodology is implemented using Cadence Incisive Unified Simulator mixed language simulator.

The combined simulation entirety is checked for long runs and compared to the SystemC AMS stand alone simulation. The XOR parity-based cryptounit is benchmarked against the refined DES/3DES models for channel effect by plotting BER versus SNR (Figure 6). The errors usually occur in burst due to intersymbol interference, that is, several adjacent bits are corrupted in 64-bit words. Figure 7 illustrates scatter plots of the recovered signal with crypto unit as a simple 9-bit XOR (no cosimulation). Figure 8 compares constellation diagrams of the filtered output for both DES and 3DES cases. Nonlinearity effect is visible in both amplitude variation and phase jitter. The radial constellations in Figures 7 and 8 indicate noncoherent frequency interference since the transceiver model transmitted data asynchronously. Figure 9 illustrates phase imbalance in the recovered symbols.

The nonlinear response of the channel causes distortion which shows several improvement options. For example, modeling the channel more realistically, using appropriate encoding scheme in addition to the cryptography or introducing multiple levels of amplitudes, for example, M-ASK modulation. On the demodulator side, the important task is to detect the bit by sampling the wave at the end of symbol interval and compare with the threshold under noise. Here the designer can experiment with the suitable value of amplitude threshold for correctly detecting all 64 bits of DES words. Similarly for improved detection coherent ASK demodulation can be employed taking into account carrier phase information. The user would introduce a PLL in the demodulator based on the constant sampling rate to detect a signal synchronized to the transmitter carrier frequency. SystemC AMS provides the essential high modeling capabilities for such tasks.

In summary, the system performance under secure DES keys can be tuned in envelope detection, receiver amplitude, selecting the carrier frequency and automatic gain control for a given data rate or SNR. Further more, intrusion attack or false authentication behavior can be modeled in the transceiver example for validating DES cryptography. The simulations enable studying ASK/OOD modulation which is susceptible to threat monitoring and radio frequency interference. Merits of other encryption methods, for example, AES, Blowfish, or Rijndael can similarly be evaluated.

```
/* daemon (server) */

daemon_parent_pid();
nbr_of_bytes = read(new_sock, buf, sizeof buf-1);
validate_messages();
(token, crypt_cmd, rate, fd) = extract_data(buf);
fork_calls();
execv(NCVHDL, opts, args, testbench.vhd, des3.vhd);
execv(NCELAB, opts, args, generic (init_vector,
key1, key2, key3, token), worklib.des3:arch);
execv(NCSIM, opts, args, -input @tcl_script,
+loadvhpi VHDL2C_DLL, inst=:des3, +start=0, +stop=62,
worklib.des3:arch);
```

Algorithm 2

```
/* VHDL2C dynamic linked library (server) */

/* register call backs: simulation time, add, remove,
value change, etc. */
  EXPIT void VHDL2C_DLL() {...}
/* traverse VHDL design to get ports */
  EXPIT int extract_ports(vhpiHandleT scopeH,
  int level)
  {...}
/* assign ASCII IDs to ports */
  EXPIT int set_id (sigListP ptr) {...}
/* add callback */
  EXPIT int addValueChangeCallback(p_cb_data cbPtr)
  {...}
/* print and save q event whenever it occurs */
  EXPIT void access_vhdl(const vhpiCbDataT *cbPtr) {
sigListP array = (sigListP) cbPtr->user_data;
/* check whether the port that changed is
the required one */
if (strcmp(array->id, "+") == 0)
strcpy(enc_val, cbPtr->value->value.str);}
/* save port value and the event time */
fprintf(vcd, "# \%llu\n", time2int64(cbPtr->time));
fprintf(vcd,"b \%s  \%s \n", strtok(cbPtr->
value->value.str,"+"), array->id);
...}
/* delete callback */
  EXPIT int deleteValueChangeCallback(p_cb_data cbPtr)
  {...}
/* write to SystemC AMS client using the set socket */
  status_code = write (fd, enc_value, sizeof enc_val);
```

Algorithm 3

8. Discussion on Cosimulation Interface

8.1. Scalability. Multiple instances of the cosimulation interface can be used in the AMS description and each instance can be mapped to a daemon program. For each SystemC AMS simulation iteration, two VHDL simulations are triggered, one for TX path (encryption) and the other for RX path (decryption) in Figure 3. Correspondingly there are encryption and decryption daemons for each TX, RX path of the transceiver that generate their own child processes and have specific C access shared library. Although there is a single VHDL DES or 3DES model for both encryption and decryption, however parting the TX and RX functions with separate C interfaces and daemons serves two important purposes: it proves that the simulation framework is robust enough to handle multiple cosimulations, secondly isolating

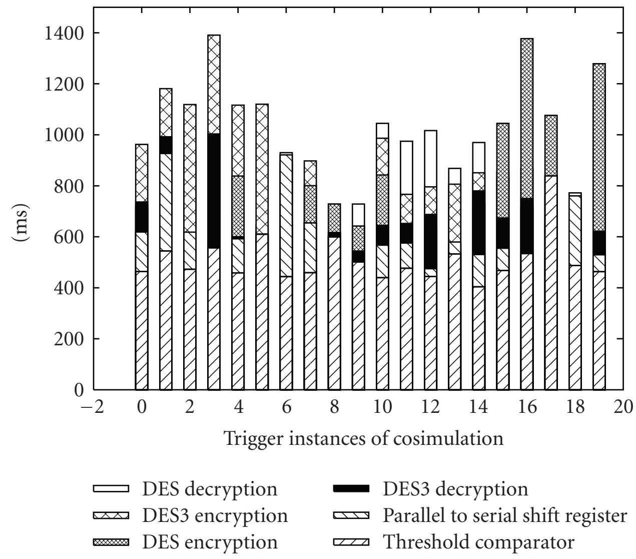


Figure 10: Coupling interface computation times.

Table 1: Degree of user modification in the framework.

Usability evaluation			
Interface element	Modification	Level	Recompilation
Strings args	always	simplest	server program
Analog control file	as needed	simpler	none
TCL and command scripts	some	simple	none
VPI/VHPI	once	difficult	only DLL

the TX and RX modularity averts contention issues of a single model being accessed concurrently by two processes and also the corruption of simulator database and simulation environment due to ambiguous dates, file modification and library loading/unloading. Using a 56-bit key and 8 bits for error detection in key generation the TX path enciphers while the RX path deciphers using the symmetric (identical) key. For faster simulation, different static shared libraries can be compiled into the simulator executable. This method generates custom simulator executables that have been augmented with user defined functions and tasks defined through the C interface. The shared libraries are produced using Cadence PLI Wizard.

8.2. Interface Computation Lag. The speed of server interface computation is related to complexity of model cosimulated (languages, tools called DLLs loaded, speed of testbench, etc.), CPU load, and network speed among others. We illustrate execution times for various models in Figure 10 (client at AMD Athlon 2.10 GHz running Ubuntu, server at Intel Xeon 3.39 GHz running Debian GNU/Linux). These nonabsolute times reflect full-scale automation including compilation, elaboration, and simulation.

8.3. Usability. Table 1 describes ease in using the cosimulation framework. The C access library ought to be generic to imported to a variety of designs, for example, by passing model instance and simulation times.

8.4. Outlook. The framework can be extended for models with analog boundaries, for example, AD converters. There are three simulation options for mixed models: (1) real-valued discrete simulations that can adequately mimic analog model behavior digitally with proper timing (top-down), (2) pure mixed-signal simulation (meet in the middle), for example, with HDL-AMS depicting accurate electrical behavior, and (3) SPICE level circuits (bottom-up). The first level is appropriate to understand the impact of implementation method on ADC resolution, dynamic range, throughput, CMRR, and quantization accuracy. For example, pipelining, SAR, Sigma-Delta, or direct conversion. Levels 2 and 3 offer further refinement into model by characterizing physical properties, for example, nonlinearities, timing (acquisition, conversion, response, settling, aperture delay), and numerical errors (gain, SNR, dither, aperture, temperature offset drift). Understanding these behaviors is vital for hw/sw codesigners who need accommodation for tolerances in their designs. The problem however with all these modeling levels is the huge amount data produced in sampling a continuous time wave which must be accumulated and segmented to be applied to the cosimulation interface for reasonable simulation speed and efficiency. These analog models will be simulated next in the framework and the methodology will be modified accordingly to handle data (setting thresholds or writing to files). The presence of floating pointing data types further exacerbates the problem of embedding a refined analog model into high level system description. Pure mixed-signal models would also demand a different solver, that is, a tool change, for example, Cadence AMS Designer, Synopsys Saber and therefore modification to simulation interface and data access mechanism as well. The SPICE netlists would have to be wrapped in HDL/HDL-AMS descriptions for enabling C level access with VHPI/VPI. The analog refinement process would require careful and qualitative evaluation because refined analog models are conservative they change the overall dynamics defined by ordinary differential or differential algebraic equations of the system in which they embed. These simulations will be inherently slow and would deal with typical convergence problem. SystemC AMS to Cadence analog cosimulation issues have been highlighted in [30].

9. Conclusion

Complex SoC designs now are being specified by executable specification in addition to conventional written requirements. SystemC AMS is a suitable simulator for this task. However SystemC AMS abstraction level is inadequate for subsystem-level, fro example, refined models. This limitation in SystemC AMS kernel can be overcome by cosimulating refined models in specialty simulators which are invoked and controlled at the module entry point of the SystemC AMS model. A mixed analog and discrete simulation framework has been presented that capitalizes on simplicity and open architecture of SystemC AMS as master simulator and its synchronization layer which eliminates the overhead of common backbone and custom synchronizing schemes typical in simulator coupling.

The framework has been demonstrated by replacing simple parity-based cryptographic unit with a refined (RTL) model of DES encryption standard, cosimulated with Cadence Incisive Unified Simulator. To evaluate robustness the framework is applied for two instances of refinement in a single system-level model. The simulation results confirm that the framework can handle complex HDL models because in absence of noise the cosimulated HDL models of encryption and decryption match pure SystemC AMS simulation which was of abstract parity checker. The framework uses C-based interfaces and can be used for any EDA vendor's tool compliant of standard C interface, for example, Synopsys and Mentor Graphics. There is capacity in the framework to simulate analog models of HDL-AMS and SPICE wrapped around C interface. The simulation interfaces are abstract written by designers who are not application programmers. The refined models are inserted directly in the executable specification and then analyzed. For SoC designers this is an enormous advantage because implementation specific details and performance can be viewed at the system definition. The methodology aims for meriting system concepts, exploring architectures, and bridging the implementation gap.

References

[1] A. Herrholz, F. Oppenheimer, P. A. Hartmann, et al., "The ANDRES project: analysis and design of run-time reconfigurable, heterogeneous systems," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '07)*, pp. 396–401, August 2007.

[2] E. S. Morales, G. Zucchelli, M. Barnasconi, and N. Jugessur, "Novel methodology for functional modeling and simulation of wireless embedded systems," *EURASIP Journal on Embedded Systems*, vol. 2008, Article ID 171358, 9 pages, 2008.

[3] S. Kajtazovic, C. Steger, A. Schuhai, and M. Pistauer, "Automatic generation of a coverification platform," in *Applications of Specification and Design Languages for SoCs*, pp. 187–203, Springer, 2006.

[4] C. Haubelt, J. Falk, J. Keinert, et al., "A systemC-based design methodology for digital signal processing systems," *EURASIP Journal on Embedded Systems*, vol. 2007, Article ID 47580, 22 pages, 2007.

[5] M. Lukasiewycz, M. Glaß, C. Haubelt, and J. Teich, "Efficient symbolic multi-objective design space exploration," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '08)*, pp. 691–696, March 2008.

[6] T. Kogel, M. Doerper, T. Kempf, et al., "Virtual architecture mapping: a systemC based methodology for architectural exploration of system-on-chip designs," in *Proceedings of the International Workshop on Computer Systems: Architectures, Modeling, and Simulation (SAMOS '03)*, pp. 138–148, July 2003.

[7] J. Gong, D. D. Gajski, and S. Bakshi, "Model refinement for hardware-software codesign," *ACM Transactions on Design Automation of Electronic Systems*, vol. 2, no. 1, pp. 22–41, 1997.

[8] A. D. Pimentel, C. Erbas, and S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 99–111, 2006.

[9] J. Zimmermann, O. Bringmann, J. Gerlach, F. Schaefer, and U. Nageldinger, "Holistic system modeling and refinement of interconnected microelectronic systems," in *Proceedings of the MARTE Workshop*, Munich, Germany, 2008.

[10] T. Kogel, M. Doerper, A. Wieferink, et al., "A modular simulation framework for architectural exploration of on-chip interconnection networks," in *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (ISSS '03)*, pp. 7–12, October 2003.

[11] D. Mathaikutty and S. Shukla, "SoC design space exploration through automated IP selection from systemC IP library," in *Proceedings of the IEEE International Systems-on-Chip Conference (SOC '07)*, pp. 109–110, September 2007.

[12] L. Bossuet, W. Burleson, G. Gogniat, V. Anand, A. Laffely, and J.-L. Philippe, "Targeting tiled architectures in design exploration," in *Proceedings of the International Symposium on Parallel and Distributed Processing*, pp. 1–8, April 2003.

[13] M. Monton, A. Portero, M. Moreno, B. Martinez, and J. Carrabina, "Mixed SW/systemC SoC emulation framework," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE '07)*, pp. 2338–2341, June 2007.

[14] J. Lee and S.-C. Park, "Hardware architecture exploration of IEEE 802.11n receiver using SystemC transaction level modeling," in *Proceedings of the International Conference on Advanced Communication Technology (ICACT '07)*, vol. 3, pp. 1707–1710, February 2007.

[15] F. Gioulekas, M. Birbas, N. Voros, G. Kouklaras, and A. Birbas, "Heterogeneous system level co-simulation for the design of telecommunication systems," *Journal of Systems Architecture*, vol. 51, no. 12, pp. 688–705, 2005.

[16] K. Popovici, X. Guerin, L. Brisolara, and A. Jerraya, "Mixed hardware software multilevel modeling and simulation for multithreaded heterogeneous MPSoC," in *Proceedings of the International Symposium on VLSI Design, Automation and Test (VLSI-DAT '07)*, pp. 1–4, April 2007.

[17] K. Gruttner, F. Oppenheimer, W. Nebel, F. Colas-Bigey, and A.-M. Fouilliart, "SystemC-based modelling, seamless refinement, and synthesis of a JPEG 2000 decoder," in *Proceedings of the Design, Automation and Test in Europe (DATE '08)*, pp. 128–133, March 2008.

[18] T. Stefanov, P. Lieverse, E. Deprettere, and P. van der Wolf, "Y-chart based system level performance analysis: an M-JPEG case study," in *Proceedings of the Workshop on Embedded Systems (PROGRESS '00)*, pp. 129–140, October 2000.

[19] M. Damm, J. Haase, C. Grimm, F. Herrera, and E. Villar, "Bridging MoCs in systemC specifications of heterogeneous systems," *EURASIP Journal on Embedded Systems*, vol. 2008, Article ID 738136, 16 pages, 2008.

[20] A. Vachoux, C. Grimm, and K. Einwich, "SystemC-AMS requirements, design objectives and rationale," in *Proceedings of the Design, Automation and Test in Europe (DATE '03)*, pp. 388–393, 2003.

[21] A. V. Christoph Grimm, M. Barnasconi, and K. Einwich, "An introduction to modeling embedded analog/mixed-signal systems using SystemC AMS extensions," Open SystemC Initiative whitepaper, June 2008.

[22] A. Vachoux, C. Grimm, and K. E. Fraunhofer, "Analog and mixed signal modelling with SystemC-AMS," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '03)*, vol. 3, pp. 914–917, May 2003.

[23] K. Einwich, "Application of systemC/systemC-AMS for the specification of a complex wired telecommunication system," in *Proceedings of the Forum on Specification and Design Languages (FDL '05)*, pp. 49–60, September 2005.

[24] M. Vasilevski, F. Pecheux, N. Beilleau, H. Aboushady, and K. Einwich, "Modeling and refining heterogeneous systems with SystemC-AMS: application to WSN," in *Proceedings of the Design, Automation and Test in Europe (DATE '08)*, pp. 134–139, March 2008.

[25] J. Denoulet, M. Alassir, O. Romain, A. Suissa, and P. Garda, "Modelling field bus communications in mixed-signal embedded systems," *EURASIP Journal on Embedded Systems*, vol. 2008, Article ID 134798, 11 pages, 2008.

[26] K. Caluwaerts and D. Galayko, "SystemC-AMS modeling of an electromechanical harvester of vibration energy," in *Proceedings of the Forum on Specification and Design Languages (FDL '08)*, pp. 99–104, September 2008.

[27] E. Markert, M. Dienel, G. Herrmann, D. Mueller, and U. Heinkel, "Modeling of a new 2D acceleration sensor array using systemC-AMS," *Journal of Physics: Conference Series*, vol. 34, no. 1, pp. 253–257, 2006.

[28] E. Markert, M. Dienel, G. Herrmann, and U. Heinkel, "SystemCAMS assisted design of an inertial navigation system," *IEEE Sensors Journal*, vol. 7, no. 5, pp. 770–777, 2007.

[29] K. Einwich, J. Bastian, C. Clauß, U. Eichler, and P. Schneider, "SystemC-AMS extension library for modeling conservative nonlinear dynamic systems," in *Proceedings of the Forum on Specification and Design Languages (FDL '06)*, pp. 113–118, September 2006.

[30] Y. Zaidi, C. Grimm, and J. Haase, "Analog behavior refinement in system centric modeling," in *Proceedings of the IEEE International Behavioral Modeling and Simulation Workshop (BMAS '09)*, pp. 31–36, September 2009.

[31] T. Maehne and A. Vachoux, "Supporting dimensional analysis in SystemC-AMS," in *Proceedings of the IEEE International Behavioral Modeling and Simulation Workshop (BMAS '09)*, pp. 108–113, September 2009.

[32] M. Damm, C. Grimm, J. Haase, A. Herrholz, and W. Nebel, "Connecting SystemC-AMS models with OSCI TLM 2.0 models using temporal decoupling," in *Proceedings of the Forum on Specification and Design Languages (FDL '08)*, pp. 25–30, 2008.

[33] K. Einwich, "SystemC-AMS Reference Manual, Version 1.0," OSCI, April 2005.

[34] C. Meise and C. Grimm, "A systemC based case study of a sensor application using the BeCom modeling methodology for virtual prototyping," in *Proceedings of the 17th Symposium on Integrated Cicuits and Systems Design (SBCCI '04)*, pp. 242–247, September 2004.

[35] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, no. 3, pp. 131–145, 1983.

[36] "DES IP module," http://www.tetraedre.com/advanced/tcdg.php.

[37] F. Regazzoni, T. Eisenbarth, J. Grobschadl, et al., "Power attacks resistance of cryptographic s-boxes with added error detection circuits," in *Proceedings of the Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT '07)*, pp. 508–516, September 2007.

[38] Y.-B. Lin and E. D. Lazowska, "A study of time warp rollback mechanisms," *ACM Transactions on Modeling and Computer Simulation*, vol. 1, no. 1, pp. 51–72, 1991.

[39] D. E. Martin, R. Radhakrishnan, D. M. Rao, M. Chetlur, K. Subramani, and P. A. Wilsey, "Analysis and simulation of mixed-technology VLSI systems," *Journal of Parallel and Distributed Computing*, vol. 62, no. 3, pp. 468–493, 2002.

[40] Y. Wu, J.-H. Li, and H.-B. Yang, "Cancellation strategy in rollback mechanism," *Journal of Shanghai University*, vol. 9, no. 6, pp. 501–505, 2005.

[41] P. J. Mosterman and J. E. Ciolfi, "Using interleaved execution to resolve cyclic dependencies in time-based block diagrams," in *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC '04)*, pp. 4057–4062, December 2004.