

High-Performance Mixed-Signal ESL Design of a Magneto Resistive Sensor Application

Martin Barnasconi, NXP Semiconductors, The Netherlands (martin.barnasconi@nxp.com)

Sumit Adhikari, NXP Semiconductors, Germany (sumit.adhikari@nxp.com)

Abstract—The application of Electronic System-Level (ESL) design methodologies and the creation of virtual prototypes has become an accepted approach in digital-centric hardware/software design teams for architecture exploration, software development and performance analysis. However, these abstract system-level models developed during the ESL design phase often lack accurate analog/mixed-signal (AMS) behavior, which means that the impact of analog elements is often excluded at the system-level. In order to design and verify a heterogeneous system architecture containing analog, digital, and software functionality, the application of modern languages and advanced ESL top-down design methodologies becomes fundamental. This paper presents the use of SystemC AMS, standardized as IEEE Std 1666.1-2016, in combination with SystemC and other C++ libraries, to enable high-performance mixed-signal system-level design of a Magneto Resistive Sensor application. It will demonstrate an efficient top-down design and modelling approach to create accurate mixed-signal descriptions, while keeping high simulation speed.

Keywords—SystemC; SystemC AMS; IEEE Std 1666-2011; IEEE Std 1666.1; Analog/Mixed-Signal, Electronic System Level; Magneto Resistive Sensor.

I. INTRODUCTION

The creation of virtual prototypes has become a well-known approach in an Electronic System Level (ESL) design flow, primarily to facilitate software development and hardware/software co-design and co-verification. Furthermore, abstract system-level descriptions are created for performance analysis and architecture exploration, as the growing complexity of system architectures demand a good understanding of the interplay between the various components in such system and the potential bottlenecks between them. However, these use cases are often digital-centric and only focus on the functionality and performance of digital hardware and software components. Today's embedded systems contain a tighter interaction between sensors, actuators, or RF interfaces and the digital HW/SW processor or digital signal processing (DSP) subsystem. In addition, the industry demands more robust system architectures to address the growing demand of functional safety and security requirements.

This means new design abstraction methods and modeling approaches should be applied to include the AMS functionality and performance in the architecture design phase. The objective is to introduce a seamless top-down refinement flow for AMS system-level design, where algorithm design, system architecture concepts and implementation-specific properties are tightly connected, covering the left side of the V-model, see Figure 1.

In this paper, the benefits of using SystemC, SystemC AMS, and various C++ libraries are presented, as a practical and compelling model-based ESL design refinement flow for mixed-signal systems.

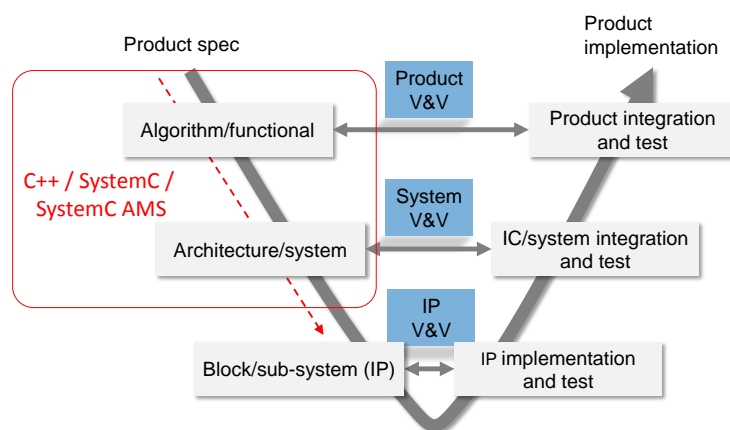


Figure 1. V-model and ESL design refinement flow

II. SYSTEMC ANALOG MIXED SIGNAL EXTENSIONS

SystemC is a language build in C++ and standardized as IEEE Std 1666-2011 [1]. It is also made available as a C++ class library reference implementation, which has been re-licensed just recently by the Accellera Systems Initiative under Apache License, version 2.0 [2]. SystemC offers a discrete-event based simulation approach, complemented with transaction-level modeling concepts, to describe digital computation and communication.

The analog/mixed-signal extensions for SystemC are defined as a separate language standard, SystemC AMS, released earlier this year as IEEE Std 1666.1-2016 [3]. The SystemC AMS language definition offers advanced modeling concepts based on timed data-flow (TDF) semantics, resulting in more efficient system simulations compared to discrete-event and real-number-modeling approaches. The data-flow engine supports multi-rate capabilities for advanced analog and digital signal processing. Furthermore, dedicated classes are defined for continuous-time simulation of linear signal-flow (LSF) building blocks and analog circuits based on electrical linear network (ELN) primitives. These three different modeling approaches support AMS modeling at different levels of abstraction, as shown in Figure 2. The abstraction levels distinguish discrete-time from continuous-time behavior and non-conservative from conservative descriptions. Discrete-time modeling is particularly suited for signal-processing-dominated applications for which signals are naturally (over) sampled. If signals cannot be sampled, the analog behavior can be described as a continuous-time function, such as by describing the system as a set of differential and algebraic equations (DAEs). A proof-of-concept simulator implementation of SystemC AMS is available under Apache 2.0 license [4].

One of the biggest advances of using C++, SystemC and SystemC AMS for ESL design and modeling is the availability of a rich set of advanced C++ libraries and utilities. For example, the use of BOOST [6] and the GNU Scientific Library (GSL) [7] have shown to be indispensable in the creation of complex and enhanced mixed-signal system-level models. For example, the statistical and Monte-Carlo library elements have been used heavily to analyze the reliability and robustness of the system architecture, which will be discussed in the next section.

Since the SystemC language is supported in most commercial EDA tools, integration of SystemC AMS models in mixed-signal IC-centric design flows is easy. For this, both the SystemC AMS model as well as its underlying simulation engine are exported as shared library (or incorporated as source files) and included in the targeted simulation environment. Recent collaborations demonstrated how commercial EDA tooling can be made more ‘SystemC AMS aware’, introducing interactive tracing and debugging capabilities [8]. In this way, the SystemC-AMS-based ESL design flow becomes well connected to the IC implementation flow, which is essential in our design refinement methodology.

The next section will further elaborate on the applied ESL design refinement methodology for AMS systems.

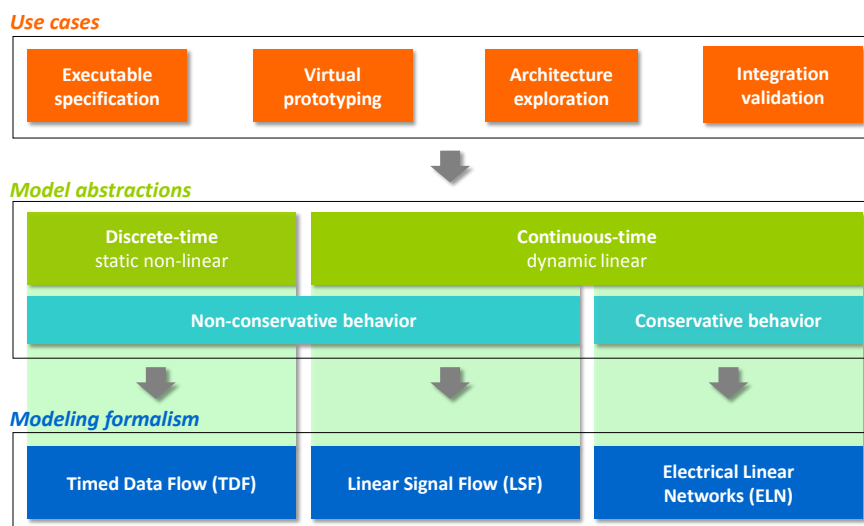


Figure 2. SystemC AMS model abstractions and modeling formalisms (from: [5])

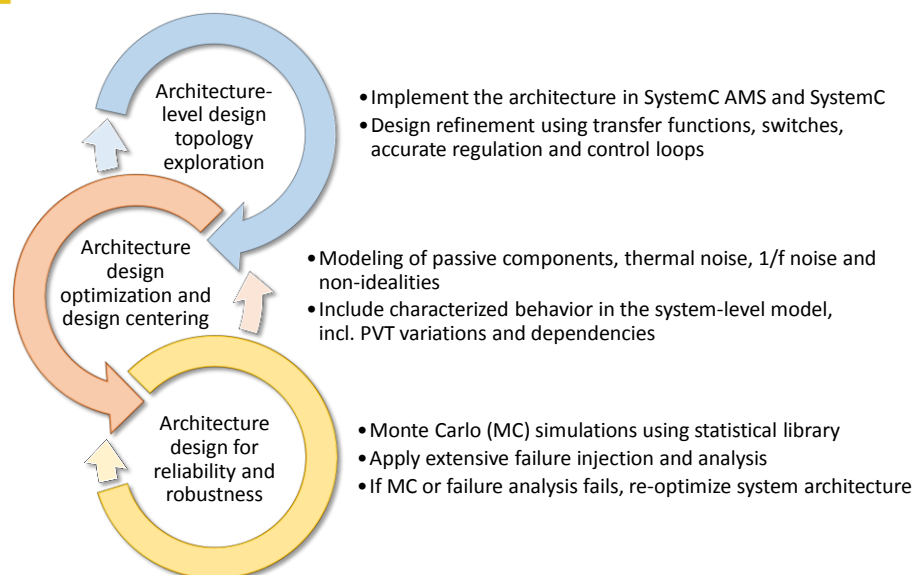


Figure 3. ESL design refinement methodology for high-performance mixed-signal systems

III. ESL DESIGN REFINEMENT METHODOLOGY FOR HIGH-PERFORMANCE MIXED-SIGNAL SYSTEMS

The ESL design refinement methodology for high-performance mixed-signal systems described in this paper consists of the following three consecutive steps, as shown in Figure 3:

- A. Architecture-level design topology exploration
- B. Architecture design optimization and design centering
- C. Architecture design for reliability and robustness

A. Architecture-level design topology exploration

The first step in the ESL design refinement flow is to define the initial design topology, resulting in an architectural composition of functional units and architecture blocks using well-defined interfaces. Note that these interfaces are not necessarily digital signal-level or bus interfaces; dedicated analog signal-level interconnect may exist at the highest level of the architecture definition. The system architecture is implemented in SystemC and SystemC AMS.

Table I shows which functions, behavior and properties are typically implemented in SystemC and/or SystemC AMS.

Table I. Typical functionality, behavior, and properties implemented in SystemC and/or SystemC AMS

<i>SystemC</i>	<i>SystemC AMS</i>
State machines	Signal sensing, amplification and mixing
Digital protocols (e.g. I2C, SPI, ...)	A/D and D/A conversion
Digital filters	Digital filters
DSP algorithms/functions	DSP algorithms/functions
Processor instruction set model	Power supply ripple
Register interfaces	Continuous-time filtering
Calibration and control algorithms	Noise contributions (white noise, 1/f noise, ...)
Transaction-level (TLM) communication	Non-linearities
FIFO's	DC offsets
...	Saturation effects
	Impedance mismatches
	AC characteristics
	Charge/discharge effects
	...

The main objective of the design topology exploration phase is to verify if the product specification and required functionality can be realized by using the intended system architecture. For high-performance mixed-signal systems, evaluation of the functionality only is not sufficient. Also analog performance needs to be studied, which requires modeling of transfer functions, switches, and accurate regulation and control loops. Note that capturing this analog behavior is done at an abstract functional system-level, to keep simulation speed high.

B. Architecture design optimization and design centering

The next step in the design refinement process is to validate and optimize the AMS system architecture, while creating the specification of the different architecture blocks and subsystems. This means refinement of the ‘micro-architecture’ by modeling passive components, thermal noise, 1/f noise and other non-idealities. In this step it is essential to use characterized behavior in the SystemC AMS model, which is extracted from the circuit-level implementations. As such, here the top-down modeling meets the bottom-up (characterization) approach. This phase can be very elaborate, since it may include variations and dependencies related to process, voltage and temperature (PVT). Also in this phase, the analog behavior is annotated to an abstract SystemC AMS system-level model.

C. Architecture design for reliability and robustness

The last step in the process focuses on system reliability and robustness. For this purpose, Monte Carlo (MC) simulations at the system-level are executed to cover more extensive PVT parameter variations. Traditional MC simulations are done at transistor-level, which are very time-consuming and lack visibility of statistical phenomena at the system-level. The presented ESL flow offers unique capabilities to execute statistical simulations at the system-level, by using a statistical library (introduced in section II)

Furthermore, to comply with automotive functional safety standards and regulations such as ISO26262, extensive failure injection and analysis is performed. The open nature of a SystemC-based simulation framework allows the creation of dedicated component classes and ‘hooks’ on top of the existing SystemC class libraries, enabling fault injection in a non-intrusive manner. If MC or failure analysis fails, re-optimization of the system architecture (step A or B) is needed, until all specification and safety requirements are met.

In the next section, the presented top-down ESL design methodology will be applied on a Magneto Resistive Sensor application.

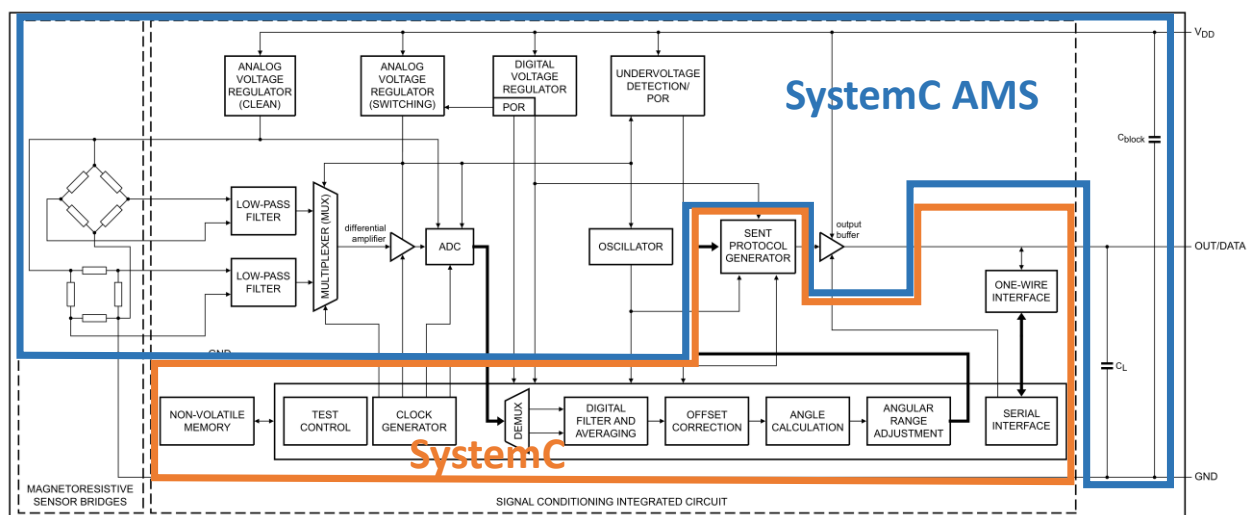


Figure 4. Functional and architecture diagram of the MR Sensor application

IV. MAGNETO RESISTIVE SENSOR APPLICATION

The presented ESL design refinement methodology is applied for the concept development of an NXP magnetic angle sensor module. It contains the following elements:

- The Magneto Resistive (MR) sensor bridges;
- A High-Performance Mixed-Signal Integrated Circuit (HPMS IC);
- Integrated capacitors.

This sensors is used for automotive applications such as position of throttle, pedal position, active suspension, electronic steering, etc.

A. Functional description

A functional and architecture diagram of a MR Sensor is shown in Figure 4. As shown in the diagram, the sensor amplifies two orthogonal differential signals from MR sensor bridges and converts them into the digital domain. The angle is calculated using the Coordinate Rotation Digital Computer (CORDIC) algorithm and transmitted in a Single Edge Nibble Transmission (SENT) frame compliant to the SAE J2716 standard. Zero angle and angular range are programmable. In addition, eight 12-bit Original Equipment Manufacturer (OEM) registers are available for customer purposes, such as sample identification. The MR Sensor comprises a Cyclic Redundancy Check (CRC) and an Error Detection and Correction (EDC) for the non-volatile memory. It also has magnet-loss and broken bond wire detection.

After multiplexing the two MR Wheatstone bridge signals and their successive amplification, the signal is converted into the digital domain by an Analog-to-Digital Converter (ADC). Further processing is done within an on-chip state machine. This state machine controls offset cancelation, calculation of the mechanical angle using the CORDIC algorithm, as well as zero angle and angular range adjustment. The SENT protocol generator converts the angular information into SENT messages that are repeatedly sent via the SENT output.

The configuration parameters are stored in a user-programmable non-volatile memory. The One Wire Interface (OWI) is used for accessing the memory. In order to protect the memory content, a lock bit can be set. After locking the non-volatile memory, its content cannot be changed anymore.

B. Product requirements of the MR Sensor

Table II presents some of the critical product requirement of the MR Sensor.

Table II. Product requirements

Parameter	Conditions	Specification			
		Min	Typ	Max	Unit
Angle resolution	programmable angular range	6		0.05	deg
Maximum angle			180		deg
Operating temperature	temperature range -40 °C to +160 °C	-40		+160	°C
Linearity error		-1		1	deg
Angular error		-1.2		1.2	deg

As shown in the table above, angular measurement errors should stay within defined limits. For example, the angular error is the difference between the mechanical angle and sensor output during a movement from α_0 to α_1 , where α_0 to α_1 are arbitrary angles within the angular range. Obviously, these errors are (ambient) temperature dependent, and the system architecture should facilitate sufficient corrections mechanisms to minimize these errors.

C. ESL modeling approach for the MR Sensor

The product application and specification briefly explained in the previous sections demand a AMS model refinement strategy for architecture selection, optimization, and validation including accurate analog behavioral modeling, as presented in section III. The analog components in the system, such as the MR sensor bridges, low-pass filters, amplifiers, A/D converters, oscillators, and supply regulators are modeled in SystemC AMS. The digital part of the system, such as CORDIC, digital filters, and digital OWI/SENT interface is modeled in SystemC.

The tight interaction between the analog and digital components in the MR Sensor architecture require the use of the dynamic time step features of SystemC AMS [9]. Listing 1 below shows a code snippet of a programmable gain amplifier (PGA) which is used in the architecture exploration phase.

```

class pga: public sca_tdf::sca_module
{
public:
    sca_tdf::sca_in<double>      in;
    sca_tdf::sca_out<double>    out;
    sca_tdf::sca_de::sca_in<bool> gain_select;

    pga( sc_core::sc_module_name nm, double f3dB1_, double dcgain1_, double f3dB2_, double dcgain2_ )
    : in("in"), out("out"), gain_select("gain_select"),
      f3dB1(f3dB1_), dcgain1(dcgain1_), f3dB2(f3dB2_), dcgain2(dcgain2_) {}

    void initialize() // initialize numerator and denominator
    {
        num1(0) = dcgain1; den1(0) = 1.0; den1(1) = 1.0 / ( 2.0 * M_PI * f3dB1 );
        num2(0) = dcgain2; den2(0) = 1.0; den2(1) = 1.0 / ( 2.0 * M_PI * f3dB2 );
    }

    void set_attributes() // TDF module supports dynamic time steps
    {
        request_next_activation( gain_select.default_event() );
        does_attribute_changes();
        accept_attribute_changes();
    }

    void processing() // time-domain implementation
    {
        double pga_out;
        if( gain_select.read() )
            pga_out = ltf_nd1( num1, den1, state, in.read(), 1.0 );
        else
            pga_out = ltf_nd2( num2, den2, state, in.read(), 1.0 );
        out.write(pga_out);
    }

    void change_attributes()
    {
        request_next_activation( gain_select.default_event() ); // next time step driven by gain_select signal
    }

private:
    sca_tdf::sca_ltf_nd ltf_nd1, ltf_nd2; // Laplace transfer functions
    sca_util::sca_vector<double> num1, num2, den1, den2; // numerator and denominator coefficients
    sca_util::sca_vector<double> state; // store equation state during reinitialization
    double f3dB1, f3dB2; // 3dB cut-off frequency in Hz
    double dcgain1, dcgain2; // DC gain
};

```

Listing 1. SystemC AMS model of a programmable gain amplifier (PGA) using the TDF dynamic time step features

As part of the second step in the ESL design refinement flow, the AMS models are augmented with non-idealities, such as introducing thermal noise following a Gaussian distribution, as shown in Listing 2.

```

#include "gaussian_rnd.h"

class pga: public sca_tdf::sca_module
{
...
void initialize()
{
...
    Rth = 1e-6; // Equivalent thermal noise
    Temp = 290.0; // Temperature
    k = 1.38064852e-23; // Boltzmann constant
    fs = 8e6; // Sampling frequency
    mu = std::sqrt(4*k*Temp*Rth*fs/2.0); // Band Limited thermal noise
}

void processing() // time-domain implementation
{
    double pga_out;
    if( gain_select.read() )
        pga_out = ltf_nd1( num1, den1, state, in.read() + mu*r.get_value(), 1.0 );
    else
        pga_out = ltf_nd2( num2, den2, state, in.read() + mu*r.get_value(), 1.0 );
}
}

```

```

// file: gaussian_rnd.h
// Helper class to calculate a
// Gaussian random number using
// the GNU Scientific Library

#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>

template < typename P >
class gaussian_rnd
{
public:
    gaussian_rnd(...);
    P get_value();

private:
    const gsl_rng_type* T;
    gsl_rng* r;
};

```

```

    out.write(pga_out);
  }

private:
  ...
  gaussian_rnd<double> r;
  double Rth, Temp, k, mu, fs;
};

```

Listing 2. SystemC AMS model including non-idealities like thermal noise using the GNU Scientific Library

D. Results

Figure 5 demonstrates the digitized output of the temperature sensor part including linearization algorithm. The solitary plot shows that after correction the error in measurement is 0.9% over the temperature range from -40 degrees to 195 degrees. On the right hand side, Monte Carlo simulation output has been demonstrated over 5 sigma process parameter variation. The results are within 1% temperature error. The Monte Carlo simulation also shows that how much maximum error percentage is attainable statistically over the life-time of the product. Please note that a margin of error has been kept aside to accommodate circuit level errors.

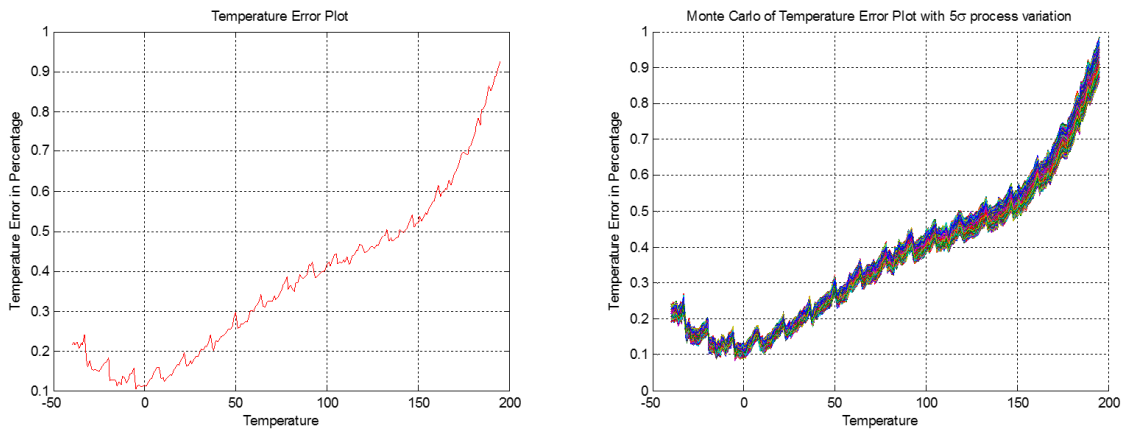


Figure 5. Typical temperature error of the MR Sensor

In Figure 6, the signal marked in red is the stable sensor output where the thermal noise is clearly seen. The yellow and green signals are two sigma delta modulator outputs where the chopping of offsets are clearly visible. Next green signal is the SENT protocol output going into the test bench where the system verification is performed.

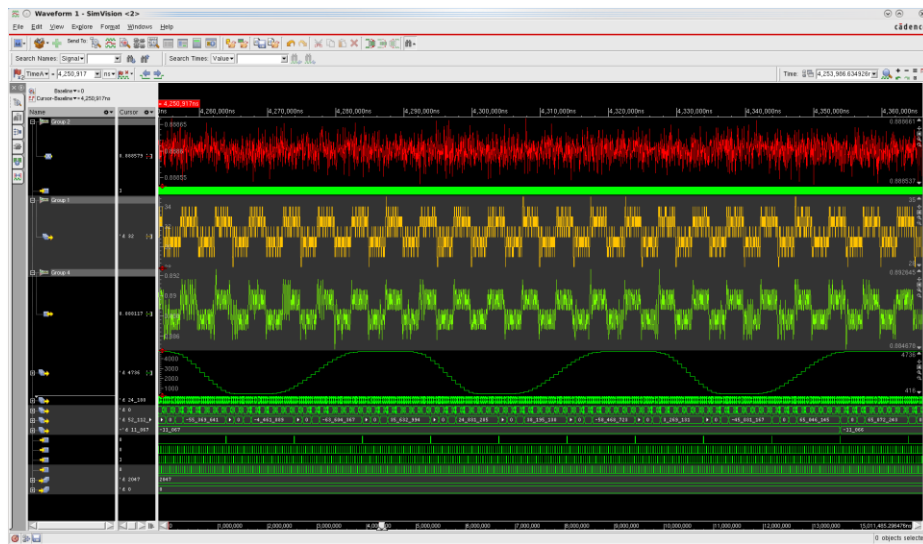


Figure 6. Mixed-signal transient simulation results with SystemC and SystemC signals

Table III below shows the typical simulation speed in the different refinement phases. During the architecture exploration phase, the most abstract models are used, resulting in the highest simulation speed. For design optimization and performance analysis, more refined AMS models with non-idealities are introduced, resulting in a speed degradation of $\sim 10\times$, but simulations still run within 10 minutes. Simulation for reliability using Monte Carlo is only done on a small portion of the input waveforms. These simulation results are considered state-of-the-art and outperform traditional HDL-language based AMS and real-number-modeling approaches. More importantly, the design space exploration step cannot be performed with these traditional modeling approaches, since simulations break due to convergence or memory allocation issues.

Table III. Typical simulation speed observed in each design refinement step

<i>Design refinement step</i>	<i>Simulation speed</i>	
	<i>Simulation time (sec)</i>	<i>Wall clock time (sec)</i>
A. Architecture-level design topology exploration	1	~ 60
B. Architecture design optimization and design centering	1	~ 500
C. Architecture design for reliability and robustness	1ms	~ 5 (per MC run)

V. CONCLUSIONS

This paper presented a novel and powerful ESL design refinement methodology and flow for high-performance mixed-signal system-level design, based on the IEEE standard languages SystemC and SystemC AMS. System-level modeling using these languages offers unique capabilities to accurately describe analog/mixed-signal behavior, while keeping a high simulation speed, which is essential for the iterative architectural exploration design process including analog performance analysis.

The ESL design refinement methodology was demonstrated for the concept development of a magnetic angle sensor module. This mixed-signal application has demanding specifications which required a SystemC-based modeling strategy to design a suitable system architecture and topology. SystemC AMS system-level models were created and refined, to enable making efficient optimizations at the system-level and to analyze the reliability and robustness of the architecture. In addition, intensive use of additional C++ libraries, for example for statistical and frequency analysis, in combination with SystemC and SystemC AMS, have been proven to be fundamental in this modeling exercise. This ESL design methodology resulted in working silicon as part of the development of the most recent family of MR Sensor products of NXP.

ACKNOWLEDGMENT

The authors wish to thank Karsten Einwich of COSEDA Technologies for his leadership to drive the SystemC AMS proof-of-concept implementation and Vincent Motel of Cadence Design Systems to facilitate integration of SystemC AMS in the Cadence Incisive simulation environment.

REFERENCES

- [1] IEEE Standards Association, IEEE Std 1666-2011, <http://standards.ieee.org/findstds/standard/1666-2011.html>
- [2] Accellera Systems Initiative, SystemC reference implementation, <http://www.accellera.org/downloads/standards/systemc>
- [3] IEEE Standards Association, IEEE Std 1666-2016, <http://standards.ieee.org/findstds/standard/1666.1-2016.html>
- [4] COSEDA Technologies GmbH, SystemC AMS Proof-of-Concept 2.1, <http://www.coseda-tech.com/systemc-ams-proof-of-concept>
- [5] Accellera Systems Initiative, Viewpoint: Analog/Mixed-Signal (AMS) Extensions for SystemC, <http://www.accellera.org/resources/articles/amsviewpoint>
- [6] BOOST libraries, <http://www.boost.org>
- [7] GNU Scientific Library, <https://www.gnu.org/software/gsl>
- [8] M. Bamasconi, S. Adhikari, K. Einwich, V. Motel. Instrumenting SystemC-AMS - Interactive Debugging & Tracing of Mixed-Signal Systems in Cadence Incisive, CDNLive 2016, May 2016
- [9] M. Bamasconi, K. Einwich, C. Grimm, T. Maehne, A. Vachoux, Advancing the SystemC Analog/Mixed-Signal (AMS) Extensions - Introducing Dynamic Timed Data Flow, September 2011, <http://www.accellera.org/resources/articles/amsdynamictdf>