Modeling and Refining Heterogeneous Systems With SystemC-AMS: Application to WSN

Michel Vasilevski*, Francois Pecheux*, Nicolas Beilleau*, Hassan Aboushady*, Karsten Einwich§
*University Paris VI, Pierre & Marie Curie LIP6-SoC Laboratory, 75252 Paris, France, § Fraunhofer IIS/EAS, Dresden, Germany
{michel.vasilevski,francois.pecheux,nicolas.beilleau,hassan.aboushady}@lip6.fr, karsten.einwich@eas.iis.fraunhofer.de

Abstract— **The paper presents a system-level approach for the modeling and simulation of a paradigmatic Wireless Sensor Network composed of two nodes using SystemC-AMS, an open-source C++ extension to the OSCI SystemC Standard dedicated to the description of heterogeneous systems containing digital, analog, RF hardware IPs as well as embedded software. The paper is composed of three parts. The first part details the modeled WSN (physical sensor, sigma-delta ADC, ATMEGA128 8-bit microcontroller running the embedded application, QPSK-based 2.4 GHz RF transceiver), presents the corresponding implementation in SystemC-AMS, and gives an insight on how multi-frequency simulation is handled in SystemC-AMS. The second part shows how to introduce several RF designer specifications (noise figure, IIP3, ...) into models and how to express them in SystemC-AMS. The third part proves that the combination of C++ and RF baseband equivalent dramatically reduces simulation time while keeping excellent accuracy and code readability. The paper concludes on the possibilities offered by this approach in terms of validation and optimization of heteregeneous systems through open-source simulation.**

I. Introduction

Needless to say that one of the great challenges of the next decade is pervasive/wireless computing. In this context, the ability to design optimal Wireless Sensor Networks is of paramount importance. To improve their competitiveness, major players in the microelectronics industry are faced with two antonymic issues : 1- the need to dramatically reduce the cost and design time of their products like SoCs or SIPs for economical reasons, 2- the lack of a unified design environment that can be used efficiently by system designers to model and simulate state-of-the-art systems (i.e. systems that encompass several research activity fields and combine on the same integrated circuit physics, analog and digital electronics, RF/micro-wave and software application). For the past 20 years, hardware description languages have been widely used to model and simulate systems belonging to various engineering fields, from digital and analog electronics to mechanics, RF and even battery cell chemistry. EDA industry proposed recently consistent modeling and simulation frameworks that allow for the description of systems from different disciplines and for the description of interactions between these systems. These frameworks use VHDL-AMS [1] [2] [3] [4] and Verilog-AMS [5] [4] as effective backbones for modeling. However, when dealing with WSN containing dozens of nodes, and with a carrier frequency of 2.4 GHz, these frameworks show rapidly their limits in terms of interoperability and simulation performance. One possible solution to the modeling and simulation of "More than Moore" multiprocessor heterogeneous systems [6] is SystemC-AMS [7] [8] [9] [10] [11], an extension to the existing library SystemC [12].

A first experimental version which will be used as a starting point for standardization has been released [13]. Figure 1, extracted from the SystemC-AMS documentation, shows the layered architecture of SystemC-AMS on top of SystemC. This architecture addresses different modeling domains, their mapping to various Models of Computation (MoC) and their sound interaction. This way, different parts of a complex heterogeneous system can be modeled and simulated within their optimal methodology and solving algorithm. For system engineering, this approach allows for a higher modeling efficiency and a faster simulation by an order of magnitude. The available SystemC-AMS prototype includes two modeling domains. The first is the multirate synchronous dataflow domain (SDF), which can be used to describe analog non-conservative (signal-flow) behaviors. In SDF, the behavior of an analog intellectual property is defined as a cluster, i.e. a set of interconnected modules with communicating input and output ports. The second domain is the conservative description, which can be used for linear electrical networks (LN_MoC). Because the application fields are restricted to linear dynamics and nonlinear static (an acceptable assumption for a wide range of communication applications), a very fast equation solver can be applied. In our application, the multirate SDF domain is of particular interest, whereby continuous time behavior of a subpart of an analog block is embedded into the dataflow module processing method named **sig_proc()**. This method is always activated if enough samples are available at the module input ports. For synchronous dataflow, the number of samples read from the input ports and written to the output ports are known before the simulation starts. This allows for an optimal static scheduling during elaboration which leads to very high simulation performance. In SystemC-AMS, the number of samples is set by attributes. Thus, if the rate attribute of an input port in a cluster module is set to 2 (using e.g. inp.**set_rate(2)**, two samples will be read per activation, and an output port rate of 3 means 3 samples are written per activation. SystemC-AMS assigns every sample a timepoint. Usually a constant time distance between samples is assumed. This sampling time must be assigned at least to one port of a dataflow cluster (using e.g. port.**set_T(1.0,SC_MS)**) and is automatically propagated to all other ports. Multi rate SDF is especially very well suited for strong oversampled systems.

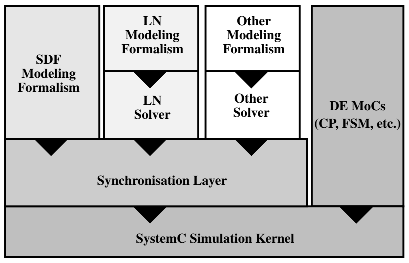

Fig. 1. SystemC-AMS layered architecture.

The figure 2 acts as the reference figure through the whole paper. It presents the peer to peer block diagram of the WSN and enlights several points of interest.

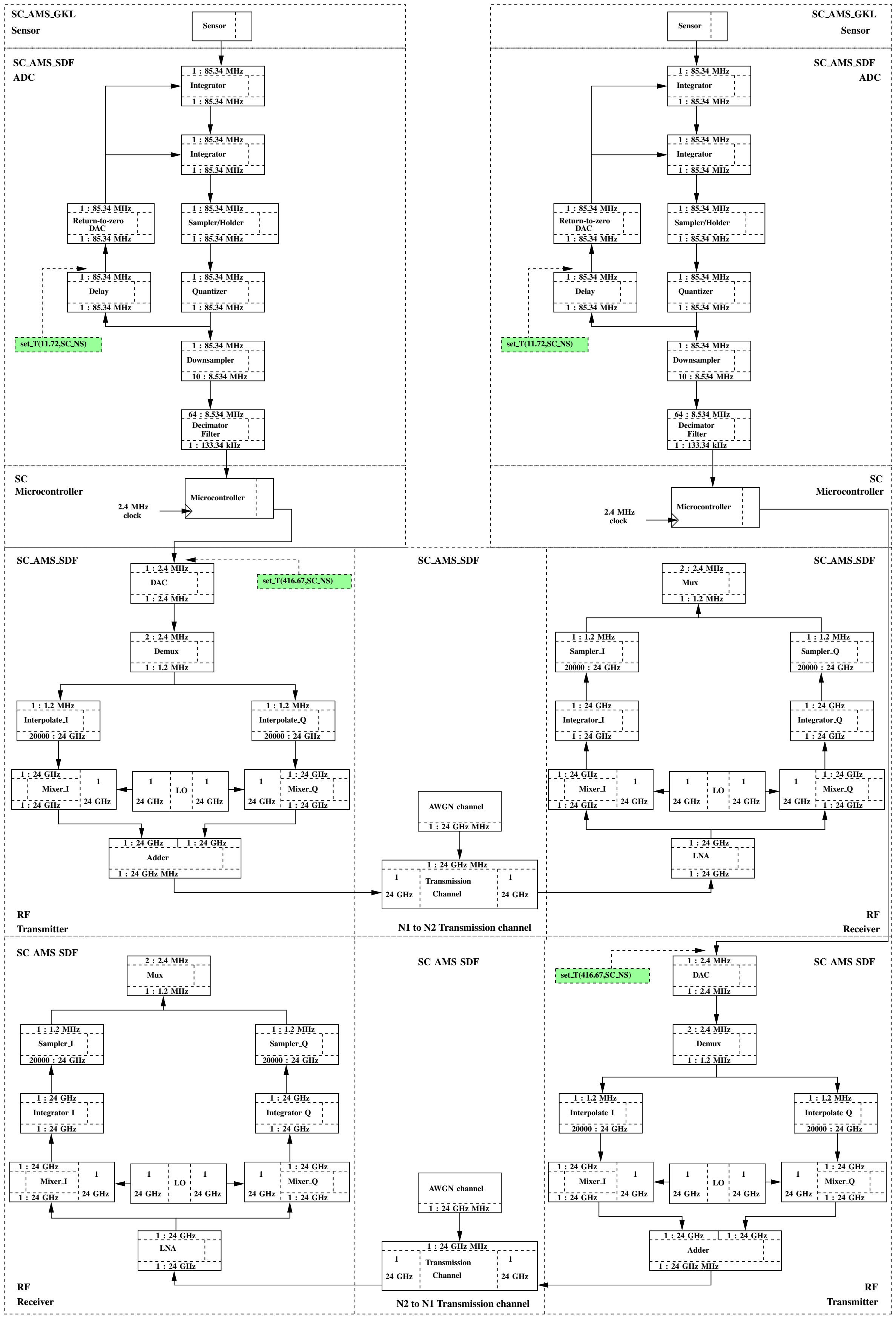

Fig. 2. Peer to peer block diagram of the modeled WSN.

There are three kinds of computation model used in this system, SystemC event-driven (SC), SystemC-AMS synchronous Dataflow (SC-AMS-SDF) and SystemC-AMS LN conservative (SC-AMS-GKL). Several clock domains are also required to simulate the system appropriately. The figure 2 also shows the representation of the system as a functional hierarchy of IPs/boxes that can consume and produce sampled data at specific frequencies and rates. For instance, the downsampler module of figure 2 has one input and one output. The input operates at 85.3 MHz and the output at 8.53 MHz. The corresponding rates indicate that 10 samples are consumed on the input for each sample produced on the output. WSN communication scheme has been greatly simplified, through the use of two independent transmission channels in full-duplex.

II. Wireless sensor network system and its direct SystemC-AMS implementation

The modeled WSN system is presented in figure 3. It consists of two independent nodes *N1* and *N2* that exchange information through a noisy 2.4 GHz communication channel. Nodes are totally equivalent from the hardware and software standpoints.

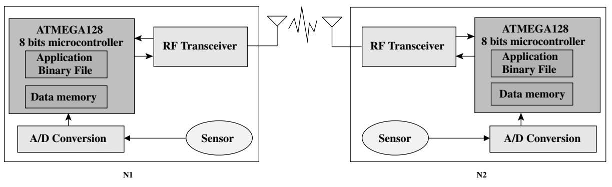

Fig. 3. The WSN, consisting of two nodes N1 and N2.

The behavior of the WSN is the following: *N1* acquires an analog measure from its physical sensor and uses an ADC to convert it into its 8-bits digital equivalent. The *N1* ATMEGA128 AVR microcontroller reads the 8-bits value on its GPIO ports, continuously executes AVR instructions to serialize the read data, and propagates the corresponding bitstream on a 1-bit port configured as an output. The bitstream is emitted by the *N1* RF transmitter, and is received by the *N2* RF receiver. In figure 2, it can be seen that the received bitstream is not propagated to the *N2* microcontroller, because no synchronization protocol has been implemented yet that would allow the receiving mote to identify the exact beginning of a new frame transmission. In the same time, the *N2* mote performs exactly the same work and propagates a sensor measure to *N1*. Simulation results given below take into account this full duplex behavior.

A. Sensor

The sensor part, described in figure 4, is quite unrealistic but is sufficient for our application. The current source and load resistor are modeled using the conservative view and means offered by SystemC-AMS.

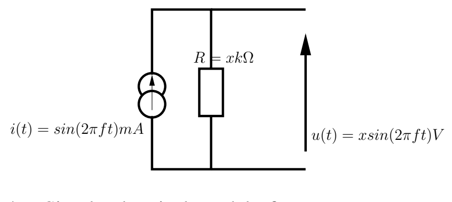

Fig. 4. Simple electrical model of a sensor.

Listing 1 shows how the design of figure 4 can actually be coded in SystemC-AMS. the **sca_elec_port** (line 6) is a conservative port and the **sca_elec_ref** represents the reference node (a node which has always the absolute voltage of 0V). Like their VHDL-AMS terminal counterparts, the **sca_elec_port**s of different elements can be connected by wires or nodes which obey the KCL (Kirchoff Current Law). The constructor **SC_CTOR** (line 17) instanciates the two SystemC-AMS linear devices (current source and resistor) and connects them according to figure 4. Because the ADC is described in the SC-AMS-SDF domain, the conservative across value (voltage) available at the sensor output is converted into its synchronous dataflow equivalent through the use of the **sca_v2sdf** SystemC-AMS primitive.

Listing 1. The sensor conservative subpart.

```
1  #ifndef WAVE_H
2  #define WAVE_H
3
4  SC_MODULE (wave)
5  {
6    sca_elec_port w1;
7    sca_elec_ref gnd;
8
9    sca_isin  *i_sin_1;
10   sca_r     *i_r_1;
11
12   void init(double a, double f){
13     i_r_1->value = a*1000;
14     i_sin_1->freq = f;
15   }
16
17   SC_CTOR (wave) {
18     i_sin_1=new sca_isin("i_sin_1");
19     i_sin_1->p(w1); // pos
20     i_sin_1->n(gnd); // neg
21     i_sin_1->ampl=0.001; // magnitude in A
22
23     i_r_1=new sca_r("r1");
24     i_r_1->p(w1);
25     i_r_1->n(gnd);
26   }
27 };
28 #endif
```

B. A/D converter

To pragmatically experiment the capabilities of the SystemC-AMS library, the ADC that converts the analog measure coming from the sensor into its digital equivalent has been implemented as a second order sigma-delta 1-bit modulator with return-to-zero feedback [14], and a decimator using a third order FIR2 [15], that can be parameterized to generate a n-bit word, as shown in the theoretical figure 5 [16].

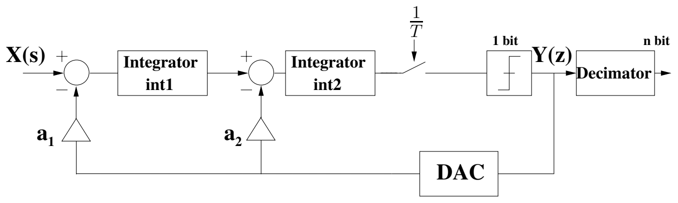

Fig. 5. Second order sigma-delta continuous-time modulator and decimator.

The boxes presented in figure 5 are all SystemC-AMS synchronous dataflow modules and belong to the ADC cluster of figure 2. For instance, the *int1* integrator module contains a **sig_proc()** function that reads the two module input values, makes a call to the SystemC-AMS implementation of the Laplace transform, and writes the result on the output. The ADC feedback loop operates at 85.3 MHz and contains a delay block required to feed the second differential inputs of each integrator with the previous value of a sample. A method closer to the physical implementation to represent the delay would be to split this delay between the ADC and the DAC and add them to their output ports using attributes. Setting the cluster sample duration time is achieved through the call to the **set_T()** function on one port of the feedback loop (at the output of the delay block in figure 2). The cluster sample duration is automatically propagated to each connected module of the cluster during model elaboration. This propagation uses the rate attribute (the default is 1) in a way that equation (1) is always fulfilled. If there are more than one sample time attribute set in a dataflow cluster, SystemC-AMS checks the consistency and may end with an error.

$$\frac{out\_sample\_time}{in\_sample\_rate} = \frac{in\_sample\_time}{out\_sample\_rate} \qquad (1)$$

The downsampler module is used to shift from a continuous-time simulation to a discrete-time simulation by keeping only 1 sample over 10. This module has been added for simulation purposes. The decimation filter has to consume 64 samples issued from the previous blocks before it can produce an 8-bit value, at a 133 KHz frequency.

C. ATMEL ATMEGA128 Microcontroller

The ATMEGA128 microcontroller belongs to the ATMEL AVR devices [17]. It is a RISC microcontroller with 16-bit wide instructions and a flash program memory of 128 Kbytes. The program executed by the microcontroller can be written in assembly language or directly in C. The AVR-GCC C/C++ compiler is freely available and can generate efficient code for this target device. From the SystemC viewpoint, the microcontroller is coded as a traditional Instruction Set Simulator (ISS) that respects the execution times of each instruction. In particular, the microcontroller SystemC/C++ class provides a member function that allows to program the flash code memory with the contents of an Intel HEX file. The AVR binary code needed to read an 8-bit value from the ADC and to serialize it takes 18 microcontroller cycles to execute. For that reason, the microcontroller clock has been set to 2.4 MHz=133 KHz*18. The bitstream frequency at the microcontroller output is also 2.4 MHz.

D. 2.4 Ghz QPSK RF transceiver

The RF transceiver is responsible for converting the digital bitstream into RF information and vice versa. It uses a coherent QPSK (Quadrature Phase Shift Keying) transmission scheme, as explained in [18] and shown in figures 6 and 7 with a f_c =2.4 GHz carrier frequency and a f_b =2.4 MHz data frequency. AWGN (Additive White Gaussian Noise) allows to take into account channel noise in the modeling of the RF communication channel and is necessary for calculating the fundamental RF characteristic BER (Bit Error Rate) with respect to SNR (Signal-to-Noise Ratio). With this nearly ideal modeling, neither the power amplifier (PA) nor the low noise amplifier (LNA) have been yet inserted in this first release of the design. Likely, the communication channel is considered as an ideal gain block with an additive white Gaussian noise (AWGN). For simulation purpose, a system-C AMS module has been added to the transmitter to interpolate the bitstream data values as it is depicted in Figure 2. With a carrier frequency of 2.4 GHz, 10 samples per period, and data frequency of 2.4 MHz, the output of the upsampler has been modified to have a rate of 20000.

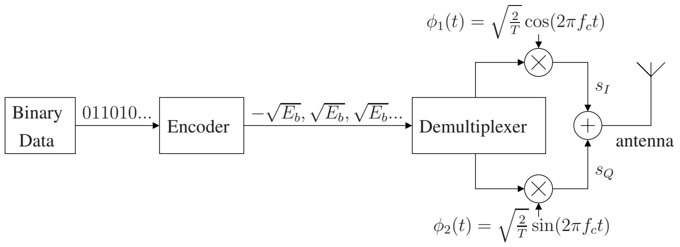
$\phi_1(t) = \sqrt{\frac{2}{T}}\cos(2\pi f_c t)$
Binary Data
011010...
Encoder
$-\sqrt{E_b}, \sqrt{E_b}, \sqrt{E_b}...$
Demultiplexer
s_I
s_Q
antenna
$\phi_2(t) = \sqrt{\frac{2}{T}}\sin(2\pi f_c t)$

Fig. 6. QPSK RF transmitter.

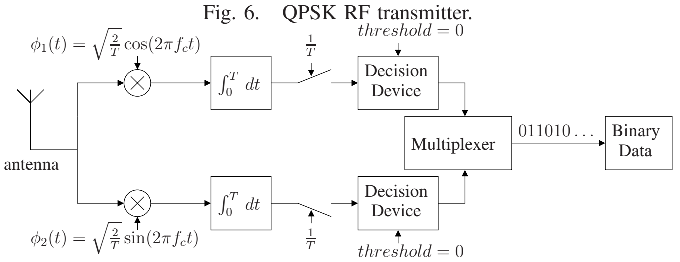

Fig. 7. QPSK RF receiver.

E. Simulated platform

The simulated platform contains about 30 files, organized hierarchically with subdirectories corresponding to subparts. The application running on each microcontroller is currently written in AVR assembly language and converted into a binary file that complies to the Intel HEX format. During the elaboration of the SystemC-AMS simulation, the two microcontrollers call their respective **initFromHEX()** function to initialize their flash code RAM. C based language for embedded application description is compiled with GNU AVR C Runtime Library. SystemC-AMS trace results can be displayed with GNU gnuplot and one can notice that all the tools (SystemC, SystemC-AMS, gnuplot) needed to obtain simulation results are totally open source.

ADC oversampling rate has been set to 64, and decimator has been configured to 8 bits. Gain values of $\Sigma\Delta$ feedback loop are specified from amplitude histogram analysis, and are respectively set to 2 and 7/6.

Simulation times, with respect to Matlab are given in table I. When possible SystemC-AMS simulations have been compared to Matlab/simulink, controlling the exact equivalent behaviour and the same number of samples used. Simulation of communication between 2 motes cannot be performed with Matlab, because of the complexity of microcontroller modeling. This problem reveals one true advantage of the SystemC-AMS simulation.

TABLE I
SIMULATION RESULTS.

	Configuration	Simu. Parameters	Matlab	SystemC-AMS
ADC	OSR=64 8 bits	1 ms 16*1024 pts	1.60s	0.93s
RF	2.4 GHz carrier freq.	416.67 μs 10^3 pts for digital part 10^7 pts for RF part	2m30.74s	54.36s
2-mote WSN	Same settings	416.67 μs	–	3m1.65s

In the RF part, an analysis has been performed to display bit error rate according to SNR variation, that needed the actual transmission of 10 Kbits. A theorical BER has been computed from AWGN characteristics and has been successfully compared to simulation results. Figure 8 shows the perfect match between simulation and theorical results. This analysis has been extended in [16] to transceiver impairments.

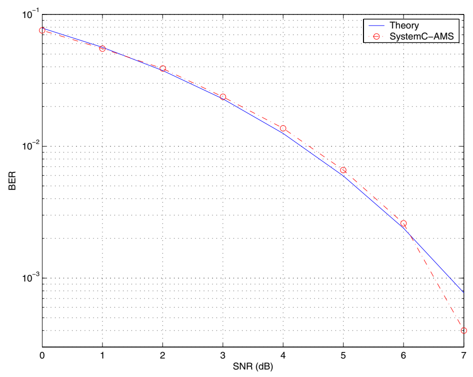

Fig. 8. Bit error rate for QPSK transmission through an AWGN channel.

III. RF Model Refinement in SystemC-AMS

The quality of the transmission scheme relies on the knowledge of the RF designer and on the refinement of the main specifications of the RF modules. Considering the theoretical model of the transceiver detailed earlier, this section, describes how to take RF specifications into account and how to express them with SystemC-AMS. The Low Noise Amplifier is used as an example.

A. LNA RF specifications

Based on the models used in [19] and illustrated in figure 9, the input parameters of the LNA are identified as the power available gain, the input and output impedances, the Noise Figure (NF) and the 3rd order Input Intercept Point (IIP3).

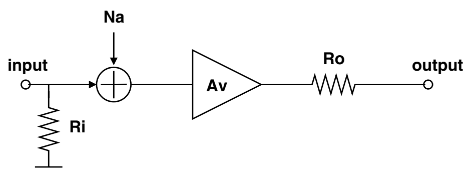

Fig. 9. RF block model

In a first step, the thermal noise of the amplifier is added to the signal. Thermal noise is given by the specified noise figure (NF):

$$N_a \quad = \quad 4KT(NF-1) \qquad (2)$$

Then the gain and non-linearities are added with a polynomial representation:

$$V_{out} = 0 + A_1 * V_{in} + A_3 * V_{in}^3 \qquad (3)$$

where A_1 is extracted from the power available gain, input resistance (R_i) and next block input resistance (R_l) with the following equation:

$$A_1 = \sqrt{\frac{G_p R_l}{R_i}} \qquad (4)$$

and A_3 is derived from the IIP3 parameter:

$$A_3 = \frac{4 * A_v}{3 * IIP3^2} \qquad (5)$$

B. SystemC-AMS implementation and simulations results

Considering these designer specifications, the implementation in SystemC-AMS of the LNA is straightforward:

Listing 2. LNA implementation

```
1  ...
2  SCA_SDF_MODULE (lna)
3  {
4    sca_sdf_in < double >in;
5    sca_sdf_out < double >out;
6    double gain_power, a1, a3, AIP3, sigma;
7    double rin, *rloadI, *rloadQ;
8
9    void init(sc_time ts, double gain_power_db,
10             double iip3, double nf,double rin,
11             double *rloadI, double *rloadQ){
12     double f  = pow(10,nf/10), N0 = 4*(f-1)*K*T*50;
13     double fs = 1/ts.to_seconds();
14     this->sigma=sqrt(N0*fs/2);
15     srand (time(NULL)); //randomize
16     this->rin=rin; this->rloadI=rloadI;
17     this->rloadQ=rloadQ; this->AIP3=undbm(iip3);
18     this->gain_power=pow(10,gain_power_db/10);
19   }
20
21   void sig_proc () {
22     double rload=
23       (*rloadI)*(*rloadQ)/((*rloadI)+(*rloadQ));
24     this->a1 =
25       sqrt(gain_power*rload/rin);
26     this->a3 =
27       a1/(3*pow(AIP3,2)/4);
28     double input = in.read()+sigma*randn();
29     out.write (a1*input-a3*pow(input,3));
30     ...
31   }
32   SCA_CTOR (lna) {}
33 };
```

The LNA module has SDF input and output ports (line 4 and 5 in the listing) that carry *double* sample values. The **init()** function is called once during model elaboration and computes the RF coefficients used throughout simulation. The **sig_proc()** function (line 21 to 31) contains the actual behavior of the LNA module. Figure 10 indicates by simulation that the RF refinements are actually taken into account.

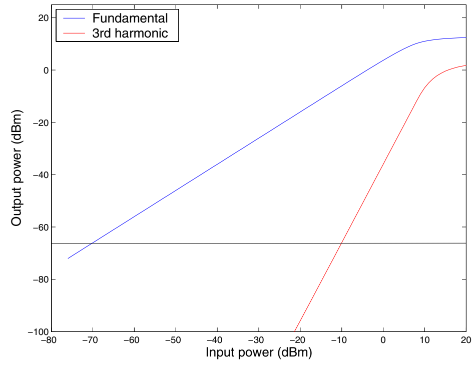

Fig. 10. Simulations results of the LNA illustrating the non-linearities and the thermal noise implementations.

IV. Baseband Models in SystemC-AMS

The standard WSN simulation used so far has naturally shown that most of the time is spent in the simulation of the RF part, with exactly 24 billion samples generated for 1 second of simulation time. To prevent this simulation time to become prohibitive, and hence to validate and optimize parts of the WSN, a common technique [20] is to identify an equivalence of the RF signal with its baseband representation to remove the carrier from the RF signal expression :

$$x(t) = DC + I_1\cos(\omega t) + I_2\cos(2\omega t) + I_3\cos(3\omega t) \\ +Q_1\sin(\omega t) + Q_2\sin(2\omega t) + Q_3\sin(3\omega t) \qquad (6)$$

In the baseband equivalent transmission scheme, the only data actually transmitted over the RF channel are the 7 coefficients of equation 6 that represent signal harmonics and their associated 2nd and 3rd order distortions, at a rate that is ten thousand times smaller than in the original simulation.

The shift from scalar representation (*double* values) to vector and matrices can be simply done with SystemC-AMS, by taking advantage of the C++ power.

As shown in listing 3, a class called BB has been defined that implements the vector, matrix and related operators. With respect to the original code, the only modification to be done is to template the **sca_sdf_in** and **sca_sdf_out** module ports with BB instead of *double*.

Listing 3. Baseband equivalent implementation

```
class BB{
 public:
   double DC,I1,I2,I3,Q1,Q2,Q3,w;
   ...
   BB operator* (double x) const{
     BB z(DC*x,I1*x,I2*x,I3*x,Q1*x,Q2*x,Q3*x,w);
     return z;
   }
   BB operator* (BB x) const{
     BB z(
 DC*x.DC+I1*x.I1/2+I2*x.I2/2+I3*x.I3/2
+Q1*x.Q1/2+Q2*x.Q2/2+Q3*x.Q3/2,
  ...
+I2*x.Q1/2+I1*x.Q2/2+DC*x.Q3,
w);
     return z;
   }
   BB operator+ (BB x) const{
     BB z(
         DC+x.DC,
         I1+x.I1, I2+x.I2, I3+x.I3,
         Q1+x.Q1, Q2+x.Q2, Q3+x.Q3,
         w
         );
     return z;
   }
};
```

TABLE II
SIMULATION RESULTS.

Simulation	SC-AMS classical simulation with refinements	SC-AMS BB equivalent RF simulation
1000 bits transmission	1m2.958s	0m0.036s
DC offset -1e5:5e3:1e5	0m19.916s	0m0.018s
Freq. offset 0:20:1e3	0m24.918s	0m0.022s
Phase mismatch 0:$\frac{\pi}{360}$:$\frac{\pi}{4}$	0m44.407s	0m0.031s

The simulation results in table 2 correspond to sections 3 and 4. As expected, simulation time has decreased by several order of magnitude.

V. Conclusion

The paper shows that the system simulation of a complete WSN that encompasses several domains is actually possible with open-source tools, with excellent accuracy. Model interoperablity and performance are obtained through the use of C++, SystemC and SystemC-AMS, and simulation times (when using state of the art RF modeling techniques) seem extremely encouraging. The complete source code is available at [21]. Ongoing research focuses on adding a real communication protocol to handle the asynchronous reception of mote data packets.

References

[1] P. Nikitin, E. Normark, C. Wakayama, and R. Shi, "VHDL-AMS modeling and simulation of BPSK trandceiver system," *IEEE International Conference on Circuits and Systems for Communications (ICCSC)*, June 2004.

[2] J. Ravatin, J. Oudinot, S. Scotti, A. Le-clercq, and J. Lebrun, "Full transceiver circuit simulation using VHDL-AMS," *Microwave Engineering*, May 2002.

[3] E.Christen and K. Bakalar, "VHDL-AMS a hardware description language for analog and mixed-signal applications," *IEEE Trans. on Circuits and Systems, part I, Vol. 46 Issue: 10, pp. 1263-1272*, Oct. 1999.

[4] F. Pecheux, C. Lallement, and A. .Vachoux, "VHDL-AMS and Verilog-AMS as Alternative Hardware Description Languages for Efficient Modeling of Multi-Discipline Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(TCAD)*, Feb. 2005.

[5] P. Frey and D. O'Riordan, "Verilog-AMS: Mixed-signal simulation and cross domain connect modules," *Proc. 2000 IEEE/ACM International Workshop on Behavioral Modeling and Simulation (BMAS), 2000, pp. 103 108.*

[6] P. Schwarz, "Physically Oriented Modeling of Heterogeneous Systems," *3rd IMACS Symposium of Mathematical Modelling (MATHMOD), Wien, 2-4 Feb. 2000, pp. 309-318 (vol1).*

[7] A. Vachoux, C. Grimm, and K. Einwich, "Towards Analog and Mixed-Signal SOC Design with SystemC-AMS," *IEEE International Workshop on Electronic Design, Test and Applications (DELTA)*, Jan. 2004.

[8] E. Markert, M. Dienel, G. Herrmann, D. Müller, and U. Heinkel, "Modeling of a new 2D Acceleration Sensor Array using SystemC-AMS," *Internationnal MEMS Conference (IMEMS)*, May 2006.

[9] A. Vachoux, C. Grimm, and K. Einwich, "Analog and Mixed Signal Modelling with SystemC-AMS," *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2003.

[10] E. Markert, G. Herrmann, and D. Müller, "System model of an inertial navigation system using SystemC-AMS," *Forum on specification and Design Languages (FDL)*, Sept. 2005.

[11] "SystemC-AMS 0.15," Oct. 2006, http://www.systemc-ams.org/documents/systemc-ams-0-15.pdf.

[12] "SystemC," http://www.systemc.org.

[13] "SystemC-AMS," http://www.systemc-ams.org.

[14] H. Aboushady, F. Montaudon, F. Paillardet, and M. M. Louerat, "A 5mW, 100kHz Bandwidth, Current-Mode Continuous-Time Sigma-Delta Modulator with 84dB Dynamic Range," *IEEE European Solid-State Circuits Conference (ESSCIRC) Florence,Italy*, 2002.

[15] H. Aboushady, Y. Dumonteix, M. Louerat, and H. Mehrez, "Efficient Polyphase Decomposition of Comb Decimation Filters in Sigma-Delta Analog-to-Digital Converters," *IEEE Trandactions on Circuits and Systems-II (TCASII)*, Oct. 2001.

[16] M. Vasilevski, F. Pecheux, H. Aboushady, and L. de Lamarre, "Modeling Heterogeneous Systems Using SystemC-AMS, Case Study: A Wireless Sensor Network Node," *IEEE International Behavioral Modeling and Simulation Conference (BMAS)*, Sept. 2007.

[17] "ATmega128 Datasheet," Oct. 2006, http://www.atmel.com/dyn/resources/prod%5Fdocuments/doc2467.pdf.

[18] simon haykin, *communication systems, 3rd ed.* Wiley.

[19] D. Leenaerts, J. van der Tang, and C. Vaucher, *Circuit design for RF transceivers*. Kluwer Academic Publishers.

[20] D. G.-W. Yee, "A design methodology for highly-integrated low-power receivers for wireless communications," Ph.D. dissertation, University of California, Berkeley, 2001.

[21] "SystemC-AMS WSN examples," http://www-asim.lip6.fr/systemc-ams.