Efficient and Refined Modeling of Wireless Sensor Network Nodes Using SystemC-AMS

Michel Vasilevski, Nicolas Beilleau, Hassan Aboushady, Francois Pecheux
Laboratoire UPMC/LIP6/SOC, Paris, France
{michel.vasilevski,nicolas.beilleau,hassan.aboushady,francois.pecheux}@lip6.fr

Abstract—**The paper introduces a model of wireless sensor network (WSN) nodes with SystemC-AMS, a C++ based language for system-level description, then presents model refinements and simulation improvements of the system. Firstly the paper details the components of a WSN node : a sensor, an ADC, a microcontroller and a RF transceiver with some introduced impairements. Secondly, it presents the implementation of RF components refinements, for a designer closer model specifications, such as power gain, noise figure, IIP3 specifications. Finally, the third part shows the simulation improvements introduced by a baseband equivalent implementation made easier with a C++ based language.**

I. Introduction

Wireless sensor network design has become an issue of reflection since a large amount of applications is developed. Modeling such systems implies to deal with heterogeneous [1] simulations in terms of signal type (digital and analog) as well as operating frequencies. An expected purpose for modeling wireless sensor network systems is to use the same language to simulate the architectural description of the system for components specifications validation, and to run several transmissions for protocol communications. Thus, architecture designers along with network designers would work in cooperation. When some languages like VHDL-AMS [2] [3] or Verilog-AMS [3] show rapidly their limits regarding simulation performance and interoperability, it is interesting to experiment such complex architecture with SystemC-AMS [4] [5] [6] to reveal its advantages.

SystemC-AMS is an extension of SystemC [7], an open source C++ based language. It introduces two models of computation, SystemC-AMS synchronous dataflow (SDF) and SystemC-AMS conservative. In addition of SystemC event-driven model of computation (for digital description), we used such SystemC-AMS views to model WSN system. The conservative view uses linear networks to make a descritption, the sensor is described in this way. The synchronous dataflow view embeds continuous-time modules into dataflow clusters. Each cluster contains modules that exchange a dataflow. It is a set of timed values that defines the signal at a certain time. When scheduled by the SystemC simulation kernel, a dataflow cluster runs at a constant simulation time step. Each module of a cluster can specify a different time step when produced sample number is different of consumed sample number. It permits to make a heterogeneous frequency operating simulation, computing an adapted number of samples according to low or high varying dataflow. Hence, SDF is specially suited for communication systems like WSN with strong oversampling. ADC and RF parts are described with SDF models. Microcontroller is described with SystemC event-driven model of computation. For more details about language implementation, some code listings of certain modules are given in [8].

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

The WSN system consist of two nodes (*N1*, *N2*) that exchange information through a 2.4 GHz communication channel. As shown in figure 1, each node contains a sensor, an ADC, a microcontroller and a RF transceiver.

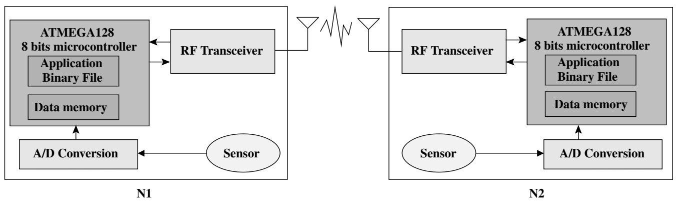


Fig. 1. The WSN, consisting of two nodes N1 and N2.

The behavior of the WSN is the following: *N1* acquires an analog measure from its sensor and uses an ADC to convert it into its 8-bits digital equivalent. The *N1* ATMEGA128 microcontroller reads the 8-bits value on its ports, continuously executes instructions to serialize the read data, and propagates the corresponding bitstream on a 1-bit port configured as output. The bitstream is emitted by the *N1* RF transmitter, and is received by the *N2* RF receiver. The received bitstream is not propagated to the *N2* microcontroller, because no synchronization protocol has been implemented yet. 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

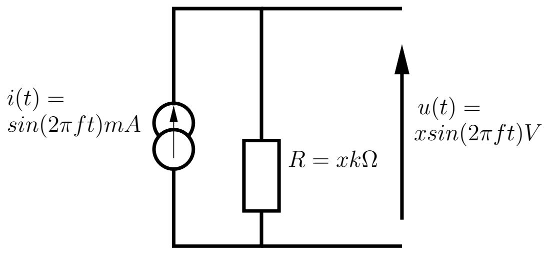


Fig. 2. Simple electrical model of a sensor.

The sensor is described in figure 2, it is a simplified model that translate analog physical variations (like temperature, pressure, wetness) to a voltage value thanks to a load resistor. The analog physical variations are modeled with a sinusoidal current source that covers a specified amplitude. The current source and load resistor are modeled using the conservative view offered by SystemC-AMS.

B. A/D converter

The ADC contains a second order sigma-delta 1-bit modulator with delayed return-to-zero feedback [9] and a decimator using a third order FIR2 filter [10] that can be parameterized to generate a n-bit word, as shown in the theoretical figure 3. Oversampling rate and number of bits produced are parameters that can be fixed following particular specifications.

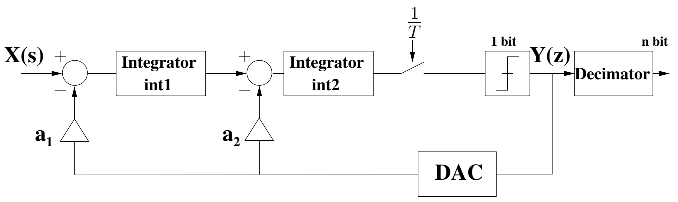


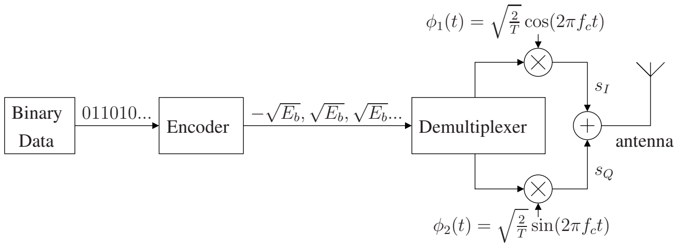
Fig. 3. Second order sigma-delta continuous-time modulator and decimator.

C. ATMEL ATMEGA128 Microcontroller

The microcontroller is ATMEGA128 [11], it is an AVR family device from ATMEL. It is a RISC microcontroller with 16-bit wide instructions and a flash program memory of 128 Kbytes. In this first approach, we used C language compiled binary file and a simplified behavior, for embedded application. Evolution of project will be oriented to TinyOS solution, an embedded operating system for wireless sensor networks [12].

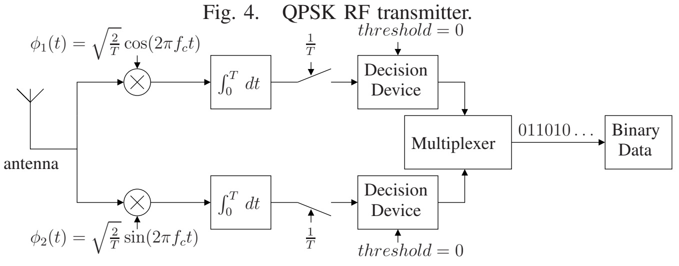
D. 2.4 Ghz QPSK RF transceiver

The RF transceiver uses a QPSK (Quadrature Phase Shift Keying) transmission, as explained in [13] and shown in figures 4 and 5 with a f_c =2.4 GHz carrier frequency and a f_b =2.4 MHz data frequency. An AWGN (Additive White Gaussian Noise) noisy channel is introduced for calculating the fundamental RF characteristic BER (Bit Error Rate) with respect to SNR (Signal-to-Noise Ratio).



$\phi_1(t) = \sqrt{\frac{2}{T}}\cos(2\pi f_c t)$
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. 4. QPSK RF transmitter.



$\phi_1(t) = \sqrt{\frac{2}{T}}\cos(2\pi f_c t)$
$threshold = 0$
$\frac{1}{T}$
$\int_0^T dt$
Decision Device
Multiplexer
011010...
Binary Data
antenna
$\phi_2(t) = \sqrt{\frac{2}{T}}\sin(2\pi f_c t)$

Fig. 5. QPSK RF receiver.

E. Simulated platform

Simulation results can be displayed with gnuplot. 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. We used a 2.4 MHz clock frequency for microcontroller, so bit rate for RF transmission is 2.4 Mbps. Carrier frequency is 2.4 GHz.

In ADC part, we simulated a variable sine amplitude input, to compute SNR characteristics of ADC and we compared to similar matlab/simulink model result.

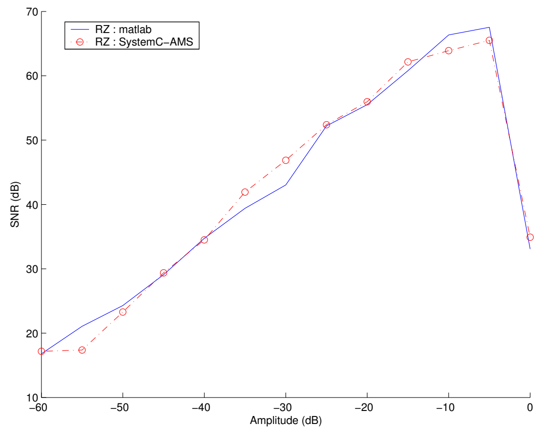


Fig. 6. SNR analysis for ADC in relation to input amplitude

In the RF part, we performed a bit error rate (BER) analysis. Figure 7 shows the match between simulation and a theorical BER from AWGN characteristics.

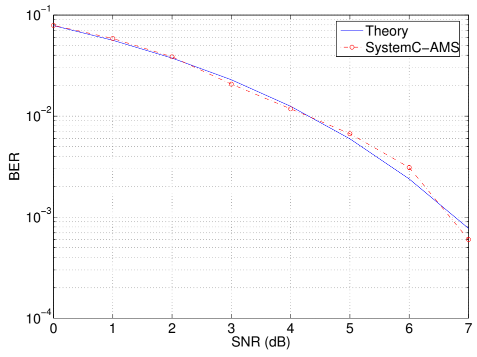


Fig. 7. Bit error rate for QPSK transmission through an AWGN channel.

Other simulations about impairements permited to display constellations according to gain mismatch, frequency offset, phase mismatch, DC offset, they are presented in figure 8.

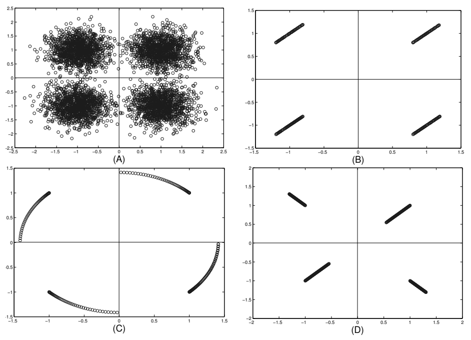


Fig. 8. Bit error rate for QPSK transmission through an AWGN channel.

Table I presents simulation times with respect to Matlab. We specified an exact equivalent configuration and used the same number of samples. 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, we are able to simulate both digital and analog models simultaneously.

TABLE I
SIMULATION RESULTS.

	Configuration	Simulation	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

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 [14] 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).

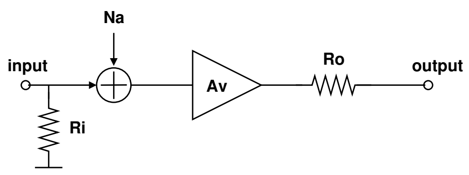


Fig. 9. RF block model

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):

$$N_a \quad = \quad 4KT(NF-1) \tag{1}$$

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

$$V_{out} = 0 + A_1 * V_{in} + A_3 * V_{in}^3 \tag{2}$$

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:

$$A_1 = \sqrt{\frac{G_p R_l}{R_i}} \tag{3}$$

and A_3 is derived from the IIP3 parameter:

$$A_3 = \frac{4 * A_v}{3 * IIP3^2} \tag{4}$$

B. SystemC-AMS implementation and simulations results

Considering these designer specifications, the implementation in SystemC-AMS of the LNA is straightforward: The LNA module has SDF input and output ports (line 4 and 5 in listing 1) that carry *double* sample values. The **init()** function (line 9 to 19) 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. The LNA module output is connected to two mixers (I and Q) for demodulation, each mixer have a load resistance. That's why we computed the equivalent resistance of such two parallel resistors for global load resistance.

Listing 1. LNA implementaion

```
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 };
```

Figures 10 present simulation results displayed with a simple plot program, like Gnuplot, Scilab or Matlab. The points to be plotted are computed thanks to a C++ FFT free library integrated in SytemC-AMS "signal analyser" module.

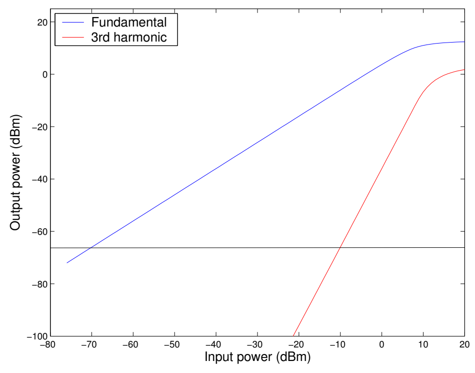


Fig. 10. Simulations results of the LNA illustrating the non-linearities and the thermal noise implementations.

IV. BASEBAND MODELS IN SYSTEMC-AMS

The standard WSN simulation used so far shows 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 too prohibitive, and hence to validate and optimize parts of the WSN, a common technique [15] [16] is to abstract the signal carrier frequency ω oscillation. Considering a signal represented by:

$$\begin{aligned} x(t) = DC + I_1\cos(\omega t) + I_2\cos(2\omega t) + I_3\cos(3\omega t) \\ + Q_1\sin(\omega t) + Q_2\sin(2\omega t) + Q_3\sin(3\omega t) \end{aligned} \quad (5)$$

In the baseband equivalent transmission scheme, the only data actually transmitted over the RF channel are the 7 coefficients of equation 5 that represent signal harmonics. The target of such representation is to plan the effect of each module behaviour to the signal harmonics, including impairements. The shift from scalar representation (*double* values) to vector (DC, I1, I2, I3, Q1, Q2, Q3 list) can be simply done with SystemC-AMS, by taking advantage of the C++ power. This power is to permit operators overloading : defining the operators function according to its operands type. Thus, rather than computing each modules behaviour in baseband equivalent, the only modification to be done is to template the **sca_sdf_in** and **sca_sdf_out** module ports with *BB* instead of *double*. As shown in listing 2, a class called BB has been defined implementing the vector and related operators.

Listing 2. 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,
  ...
  Q3*x.DC+Q2*x.I1/2+Q1*x.I2/2
              +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;
    }
};
```

Data flow time variation is significatly pull down, thereby it is possible to reduce sample simulation period and observe improved simulation timing results. The simulation results in table II correspond to sections III and IV. As expected, simulation time has decreased by several order of magnitude.

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

V. CONCLUSION

The advantages of SystemC-AMS are the capacity of interoperability and multi-frequency simulations. Moreover, a C++ based language opens the possibility to use a large choice of existent libraries or to request software programmers that don't have to learn a description language. Such asset have been exploited when including an FFT analyser to verify IIP3 specifications. Introducing baseband equivalent modeling allows to implement protocol communications such as Zigbee, adapted for wireless sensor networks. In fact, the simulation timing results are clearly improved.

REFERENCES

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

[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] 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.

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

[5] 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.

[6] 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.

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

[8] 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)*, Sep. 2007.

[9] 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.

[10] 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.

[11] "8-bit AVR Microcontroller with 128K bytes in-System programmable flash ATmega128 Datasheet," Oct. 2006, http://www.atmel.com/dyn/resources/prod%5Fdocuments/doc2467.pdf.

[12] "An open-source operating system designed for wireless embedded sensor networks." http://www.tinyos.net/.

[13] S. Haykin, *communication systems, 3rd ed.* Wiley.

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

[15] K. Kundert, "Introduction to RF Simulation and its Application," *Journal of Solid-State Circuits (JSSC), vol.34, no. 9*, Sep. 1999.

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