Modeling Heterogeneous Systems Using SystemC-AMS Case Study: A Wireless Sensor Network Node

Michel Vasilevski, Francois Pecheux, Hassan Aboushady, Laurent de Lamarre
University Paris VI, Pierre & Marie Curie
LIP6-SOC Laboratory, Paris, France
{michel.vasilevski francois.pecheux hassan.aboushady laurent.delamarre}@lip6.fr

***Abstract*— The paper presents a preliminary approach for the modeling and simulation of a simple but complete Wireless Sensor Network with two nodes using SystemC-AMS, an open-source C++ library dedicated to the description of heterogeneous systems containing digital, analog, RF hardware parts as well as embedded software. The WSN node, or mote, detailed herein consists of a physical sensor, a continuous time sigma-delta converter with its associated decimation filter, an ATMEGA128 8-bit microcontroller running the embedded application and a QPSK-based 2.4 GHz RF transceiver. The node has been designed to be interoperable with both the XBow MICAZ hardware platform and the TinyOS operating system in a near future. The paper starts with the structural description of the system as a hierarchical set of behavioural modules, then gives an insight on how multi-frequency simulation is handled in SystemC-AMS, and finally presents simulation results that are systematically compared with the Matlab reference in terms of accuracy and simulation time.**

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 the modeling. However, when dealing with WSN containing hundreds 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]. The first version has been released by Franhofer Gesellschaft EAS/IIS Dresden [13]. Figure 1, extracted from the SystemC-AMS documentation shows how the objective of multi-discipline modeling can be achieved with a set of interoperable userview layers corresponding to the aforementioned research fields.

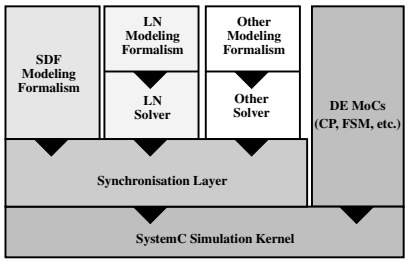

Fig. 1. SystemC-AMS layered architecture.

In practice, SystemC-AMS allows to describe mixed-signal [14] designs and currently supports two user views and their associated semantic models: conservative and multi-rate Synchronous DataFlow (SDF). For the moment, the conservative view is restricted to linear networks and does not allow the design of real analog subsytems. For the level of modeling required by this system, the multi-rate synchronous dataflow approach is of much more interest. The key idea of this approach is to embed continuous-time modules into dataflow clusters. Modules perform computation and communicate with others via directed data streams carrying time valued samples. A dataflow cluster may contain any number of dataflow modules whose execution is statically scheduled during simulation elaboration. A cluster is managed by a dedicated SystemC process that handles synchronization with the rest of the system. When scheduled by the SystemC simulation kernel, a dataflow cluster runs at a constant time step, defined by the sampling duration time assigned to one port of one of the modules and automatically propagated to others. Hence, SDF is specially suited for communication systems like WSN with strong oversampling : a SystemC-AMS module can be seen as a simple dataflow class function (always named **sig_proc()**) which, at every time step, reads its SDF inputs, computes and accumulates results, and propagates them to the SDF outputs.

II. Wireless sensor network system

The modeled WSN system, presented in figure 2, consists of two independent nodes that exchange information through a noisy 2.4GHz communication channel. Nodes are totally equivalent from the hardware standpoint, and can only be distinguished by the software application they run. The paper describes in turn the interesting parts of the system.

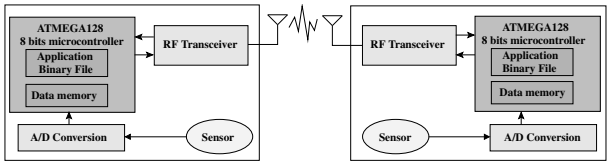

Fig. 2. The WSN, consisting of two nodes.

Structurally, a node consists of 4 parts, as shown in figure 3. One can notice that the node has four different clock domains (3.65 KHz, 8.55 MHz, 2.4 MHz, and 2.4 GHz), with frequency differences of several orders of magnitude. Section II.E shows how these domains are managed altogether by SystemC-AMS.

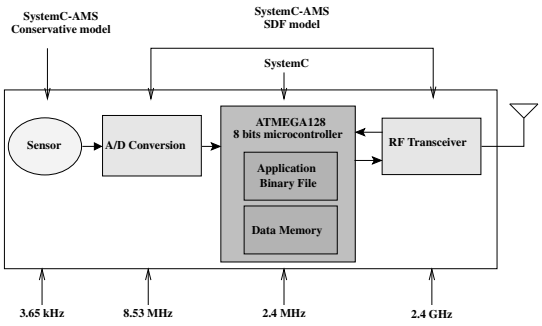

Fig. 3. Structure of a wireless sensor network mote.

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.

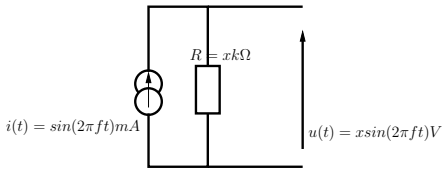

Fig. 4. Simple electrical model of a sensor.

Listing 1 shows how the design of figure 4 can actually be coded in SystemC-AMS. The **sca_elec_port** (line 6) and **sca_elec_ref** (line 7) are conservative ports that obey GKL, and are very similar to VHDL-AMS terminals. The constructor **SC_CTOR** (line 17) instanciates the two SystemC-AMS linear devices (current source and resistor) and connects them according to figure 4. To be compatible with the ADC, the conservative value is converted into its synchronous dataflow equivalent through the use of the **sca_v2sdf** SystemC-AMS construct.

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 is nothing less that a second order sigma-delta 1-bit modulator with return-to-zero feedback [15], and a decimator using a third order FIR2 [16], that can be parameterized to generate a n-bit word, as shown in figure 5. The oversampling rate can be set accordingly in order to follow particular specifications.

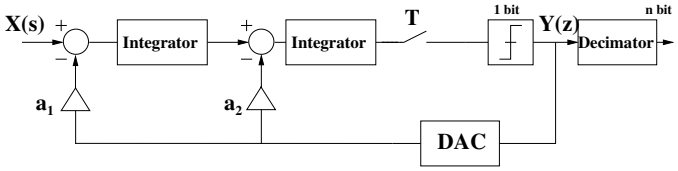

Fig. 5. Second order sigma-delta continuous-time modulator and decimator.

In the SystemC-AMS methodology, the boxes presented in figure 5 are synchronous dataflow modules and correspond to one single dataflow cluster, and therefore obey the semantic model of multi-rate synchronous dataflow graphs. Each SDF module is dedicated to the computation of a mathematical equation that consumes input data and produces output data. As an example, listing 2 gives the complete code for the integrator **integrator_sd** of the sigma-delta converter. The integrator, described as a SystemC AMS Synchronous Dataflow Module **SCA_SDF_MODULE** (line 4), takes two sdf ports as inputs, **in1** and **in2** (line 6), and generates the integrated result on **out**(line 7). The **init** member function (line 13) is called once during elaboration and allows to initialize the Laplace transform function matrix. The Laplace transform function is conveniently handled by the dedicated class **sca_ltf_nd** (line 11) and operates exactly like the VHDL-AMS corresponding construct. Each time the cluster process is triggered by the SystemC simulation kernel, the **sig_proc()** function (line 22) is called. **In1** and **in2** are read, the Laplace transform is performed, and the result written on the output **out**.

Listing 2. The complete SystemC-AMS source code for the SD integrator, using the Laplace transform SystemC-AMS native construct.

```
1  #ifndef INTEGRATOR_SD_H
2  #define INTEGRATOR_SD_H
3
4  SCA_SDF_MODULE (integrator_sd)
5  {
6    sca_sdf_in < double >in1, in2;
7    sca_sdf_out < double >out;
8
9    double fs, ai, ki;
10   sca_vector < double >NUM, DEN, S;
11   sca_ltf_nd ltf1;
12
13   void init (double a, double f, double k) {
14     DEN (0) = 0.0;
```

```
    DEN (1) = 1.0;
    NUM (0) = 1.0/3.0;
    ai=a;
    fs=f;
    ki=k;
  }

  void sig_proc () {
    out.write(
        ltf1(NUM,
             DEN,
             S,
             fs*(ai*in1.read()-ki*in2.read())
             )
        );
  }

  SCA_CTOR (integrator_sd) {}
};
#endif
```

C. ATMEL ATMEGA128 Microcontroller

The ATMEGA128 microcontroller belongs to the ATMEL AVR devices [17] [18]. It is a RISC microcontroller with 16-bit wide instructions and a flash program memory of 128Kbytes. 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. The microcontroller has been chosen for the WSN system to be compatible with the MICAZ platform commercially available, and mostly because it is directly supported by the TinyOS [19] application design environment and operating system. 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 class provides a member function that allows to program the flash code memory with the contents of an Motorola SREC or Intel HEX file. In the presented system, the 8-bit value coming from the ADC decimation filter is connected to digital port B of the microcontroller. The application software permanently reads port B, serializes the corresponding value and propagates the corresponding bitstream on pin 0 of port A. When receiving RF data, microcontroller reads pin 0 of port C.

D. 2.4 Ghz QPSK RF transceiver

The RF transceiver, presented in figures 6 and 7 is responsible for converting the digital bitstream into RF information and vice versa. It uses a QPSK (Quadrature Phase Shift Keying) transmission scheme, with a **Fc** carrier frequency and a **Fb** 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). The QPSK transmitter consists of a 1-bit D/A converter, a demultiplexer, two mixers and a signal adder. The D/A converter, also called encoder shifts voltage levels of the input bitstream following energy user specifications. Multiplied by a cosine or sine oscillation into the mixer, signals **sI** and **sQ** are combined in the adder, and the sum yields the QPSK-modulated signal. From the receiver viewpoint, the received signal is mixed with a cosine and sine oscillation and each part is integrated. The integration of cosine mixed part allows to extract the I-part of information, Q-part is similarly extracted from integration sine mixed part. A decision device digitalizes positive values to symbole '1' and negative values to symbole '0'. This block allows a hardware transmission error correction while the value received does not change its sign. The signal is finally multiplexed and get totally rebuilt. One can notice that neither the power amplifier (PA) nor the low noise amplifier (LNA) have been implemented in the design. Likely, the communication channel is considered as an ideal gain block with an additive white Gaussian noise (AWGN).

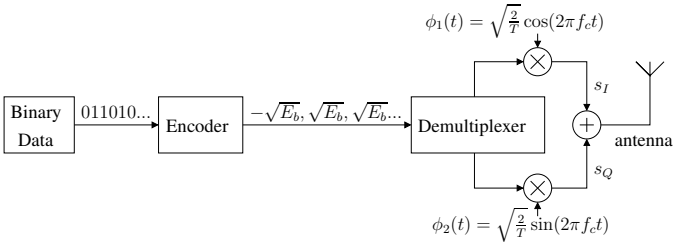

Fig. 6. QPSK RF transmitter.

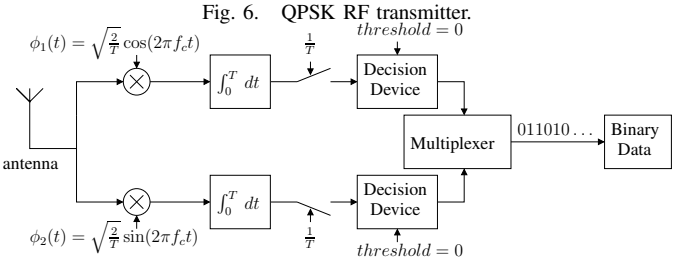

Fig. 7. QPSK RF receiver.

E. Multi-frequency modeling and module interfacing

Hierarchical design in SystemC-AMS can be achieved by means of plain SystemC modules. For example, the instanciation of the integrator module in the sigma-delta converter is obtained with the following lines of code :

```
        ...
i_integrator_1 =
  new integrator_sd ("i_integrator_1");
i_integrator_1->in1 (in);
i_integrator_1->in2 (sig_switcher_1);
i_integrator_1->out (sig_integrator_1);
        ...
```

In a SystemC-AMS design with a single clock domain, the **sig_proc()** function of each connected module of a dataflow cluster is called periodically, on the basis of the cluster sample duration time. The system designer can setup sample duration by assigning a value to a module port through the use of the **set_T()** function. The simulation sample duration is automatically propagated to each connected module of the cluster. To allow multi-frequency simulation, SystemC-AMS proposes the **set_rate()** function that can be called on a specific module port to define its relative sample rate. By default, the sample rate of each port of a dataflow cluster is 1. Sample rates control simulation sample time when a given module consumes a different number of samples that it produces. In a cluster, if the equation (1) is not verified for every module interface port, the simulator ends with an error. Sample rate is defined with respect to sample time, in order to use SystemC-AMS modules in other projects with different sample time configuration.

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

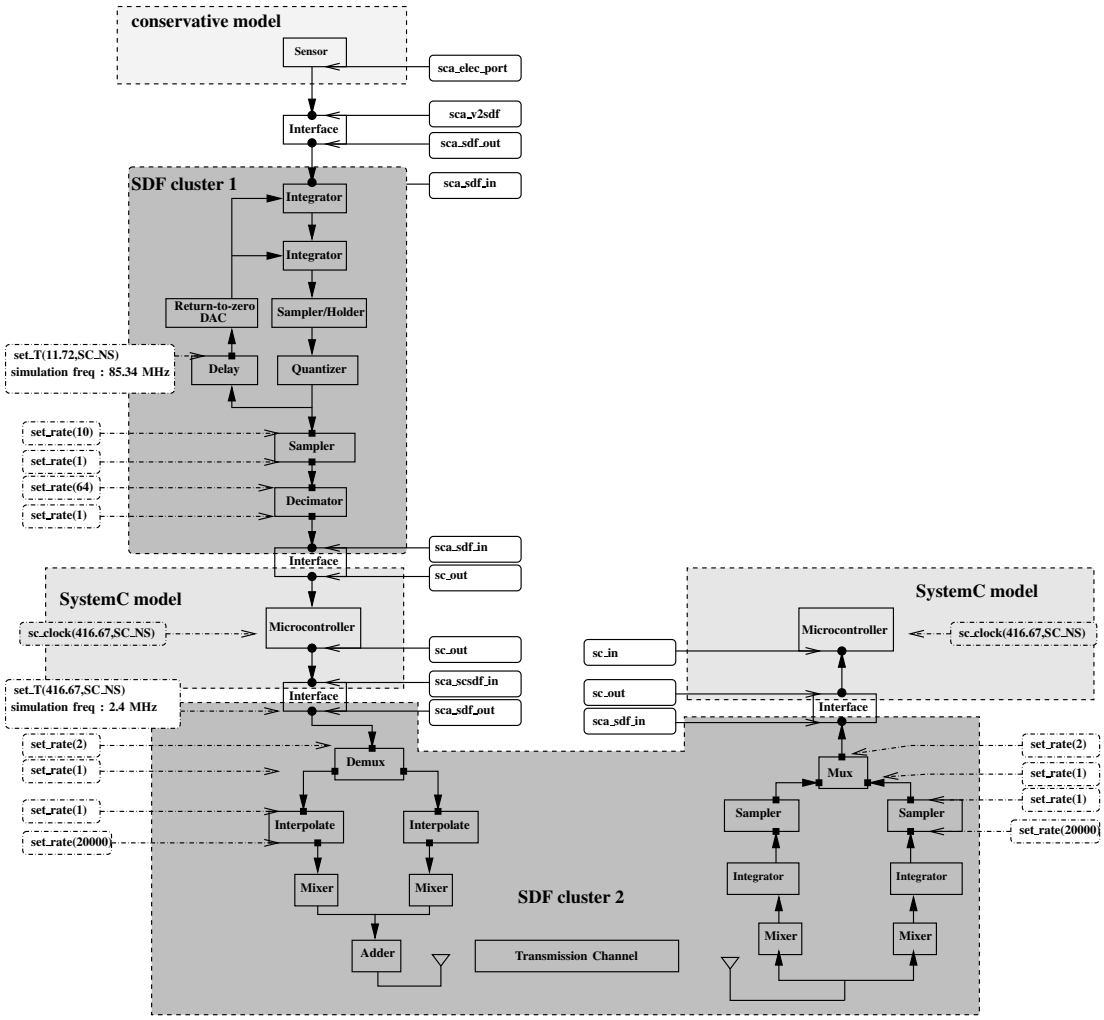

Fig. 8. Sample rate settings into WSN mote architecture.

The figure 8 presents the complete design of a standard peer to peer diagram and puts emphasis on the used semantic models(event driven, SDF or conservative), clock domains corresponding to SDF clusters and interface port kinds. When shifting from one semantic model to an other, an interface adapter is needed. The figure also shows sampling duration time (with **set_T**). Frequency of samples is set to 85.34 MHz in the delay output port, this setting is spread on the whole cluster 1 with rate adaptations : the output of sampler module operates at :

$$\frac{85.34}{10} MHz = 8.53 MHz$$

sample frequency and the output of decimator :

$$\frac{85.34}{10 * 64} MHz = 133.34 kHz$$

Regarding ADC frequency (8.53 MHz) in figure 3, the SystemC-AMS design takes 10 samples per period. That samples are not needed when the signal becomes digital, because the sampler/holder holds the sampled value during that 10 samples, the information would be identical 10 times. Thereby, we justified the sampler module implementation that changes data rate.

On cluster 2, frequency of samples is set to 2.4 MHz at the output of interface module. Like previous sampling duration computation, interpolate module puts data at its output port with a 24 GHz frequency (10 samples per period). Regarding RF frequency during the transmission in figure 3, such simulation frequency is necessary for a 2.4 GHz wave simulation.

One can notice that the A/D converter is a dataflow process with feedback. To ensure that each module of the cluster has sufficient samples on its inputs and therefore can be executed correctly, a special delay module has been added. In strongly oversampled systems like WSN, this is negligible.

III. Simulated platform

The simulated platform contains 26 files, organized hierarchically with subdirectories corresponding to subparts. The ADC oversampling rate is 64, and the decimator produces an 8-bit word. 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.

The design and simulation flow is described in figure 9. "WSN node model" is the entire project architecture written in SystemC and SystemC-AMS previously described. "Application.c" contains the C

compliant microcontroller behaviour and implements the communication protocol. C based language for embedded application description is compiled with GNU AVR C Runtime Library. Trace results can be displayed with GNU gnuplot. One can notice that all the tools needed to obtain simulation results are totally open source.

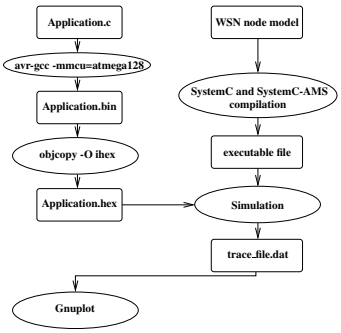

Fig. 9. Programming methodology.

IV. Simulation results

For simulation purposes, some parameters have been set up, for each component. ADC oversampling rate is 64, decimator produces 8 bits. Gain values of $\Sigma\Delta$ feedback loop are specified from amplitude histogram analysis, they are 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.

Simulation sample frequency is set to 2.4 MHz for multiplexer output or demultiplexer input, these signals are converted from or to systemC digital signals so we need just one sample per bit. According to rate settings in figure 8, simulation sample frequency for cosine mixer is 24 GHz.

The first simulation (fig. 10) images and validates RF transmission in a mote communication. The first representation (fig. 10.A) shows data to be transmitted. The next figure (fig. 10.B) represents transmitted wave through noisy channel. Finally, we can see (fig. 10.C) that data received is the same as data to be sent.

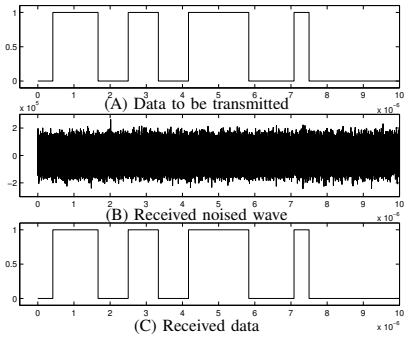

Fig. 10. Data transmission through noisy channel

After validation of behaviour, we analyse simulating durations (table I). We timed Matlab/Simulink vs. SystemC-AMS simulations, 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 the advantage of SystemC-AMS simulation : we are able to simulate both digital and analog models simultaneously, Matlab/Simulink doesn't contain microcontroller models yet.

TABLE I
Simulation results.

	Configuration	Simulation	Matlab	SystemC-AMS
ADC	OSR=64 8 bits output	1 ms 16*1024 pts for output	1.605s	0.934s
RF	2.4 GHz carrier freq.	416.67 μs 10^3 pts for μc input, 10^7 pts for RF i/o	2m30.746s	54.360s
2-mote WSN	Same settings	416.67 μs	–	4m19.572s

We verified accuracy of results with some more technical tests. We simulated a variable sine amplitude input, to compute SNR characteristics of ADC and we compared with similar matlab/simulink model result (fig. 11).

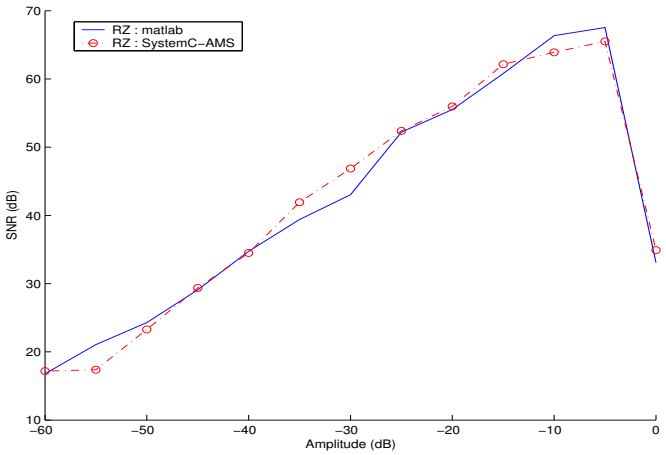

Fig. 11. SNR analysis for ADC in relation to input amplitude.

Then, we set optimal sine amplitude for maximal SNR. In fact, the better amplitude is -5 dB, so we used $a = 0.56$. We can observe frequency responses to validate ADC behaviour and to compare with Matlab results.

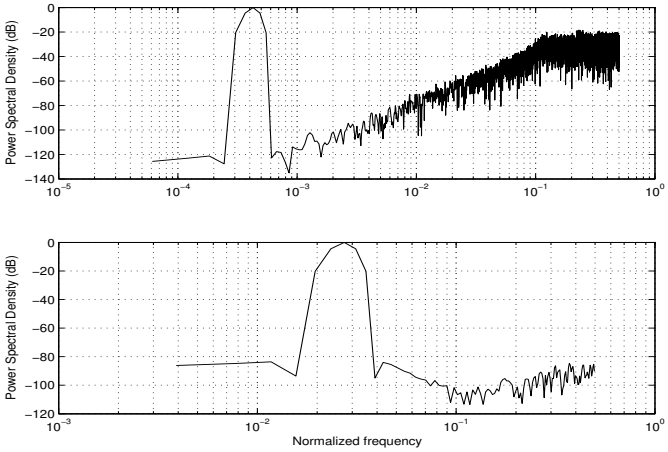

Fig. 12. Frequential response analysis of Sigma-Delta and decimator outputs.

In the RF part, we wanted to run an analysis to visualise bit error rate according SNR variation, we needed 10 Kbits. Bit error rate is the number of bad received bits divided by the number of transmitted bits. A theorical BER is computed from AWGN characteristics and is compared with simulation results. Figure 13 shows similarities between simulation and theorical results.

RF mixer module has to compute 10000*10000 sample multiplications with cosine for each I and Q part. An other transmission modeling solution, like baseband equivalent, should be used soon. In this case we will be able to abstract carrier frequency transmission and reduce simulation sample time.

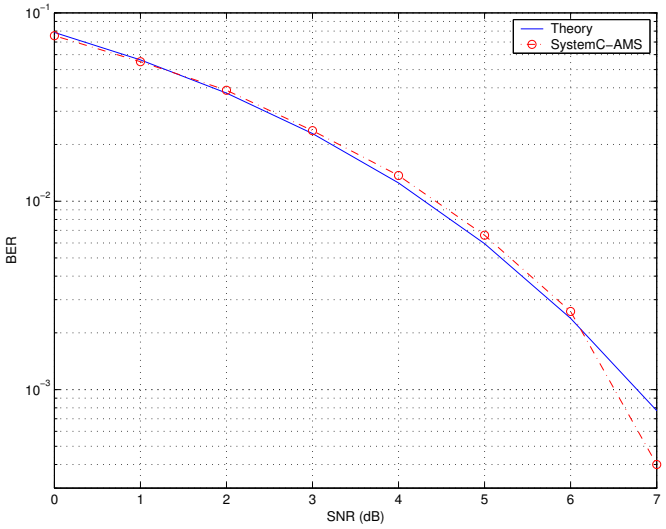

Fig. 13. Bit error rate for QPSK transmission through an AWGN channel.

Representations on figure 14 permit to image received symbol dispersion, according to different non-idealities. Those figures represent 2-bit symbols received when a non-ideality is present. We can see the value dispersion and we can remember (section II.D) that this value is corrected by decision device when the dispersion is not to strong.

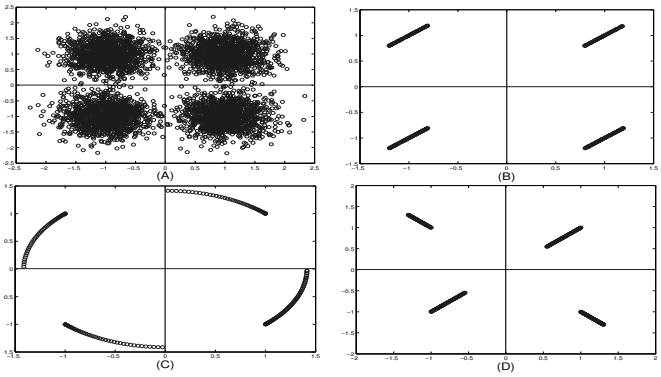

Fig. 14. Constellation of symbols received from QPSK transmission through AWGN channel (A), with a DC offset (B), frequency offset (C) and phase mismatch (D).

V. Conclusion

In the paper, we presented the prelminary results obtained from the simulation of two nodes WSN coded in C++, SystemC, and SystemC-AMS.

This work can be seen as a starting point for the modeling of a more ambitious network with other nodes perturbating the commuinication between the emitting node and the receivng node to which the message is sent. With the current platform many experiences can be performed globally or locally. In particular, we intent to introduce known approaches in the RF domain like baseband equivalent. Ongoing research will focus on the development of software using the TinyOS environment and on the automatic generation of SystemC-AMS models from subpart specifications, constraints and technological parameters. For instance, taking the ideal behaviour of a component (like an amplifier with a simple behaviour O=A.I) as a starting point, along with its structural representation in terms of transistor netlist and foundry technological parameters, we intend to automatically generate a derived SystemC-AMS model using more accurate Laplace transform construct with faithfull coefficients.

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] E. Christen, "Selected Topics in Mixed-Signal Simulation," *Forum on specification and Design Languages (FDL)*, 2002.

[15] 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.

[16] 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.

[17] "8-bit AVR Microcontroller with 128K bytes in-System programmable flash ATmega128 Datasheet," Oct. 2006, http://www.atmel.com.

[18] "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.

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