High & low-level features modelling of nodes in WSNs using SystemC

High & Low-Level Features Modelling of Nodes in WSNs using SystemC

Josie E. Rodriguez Condia
Electronics Engineering School
Universidad Pedagógica y Tecnológica de Colombia,
UPTC
Sogamoso, Colombia
josie.rodriguez@uptc.edu.co

Wilson Javier Perez Holguin
Electronics Engineering School
Universidad Pedagógica y Tecnológica de Colombia,
UPTC
Sogamoso, Colombia
wilson.perez@uptc.edu.co

Abstract — **Performing an adequate modeling of sensors in contemporary sensor networks can be difficult due to the need to include characteristics of high and low level of the entire network into a single software model. This paper presents a novel approach for modeling the nodes in a sensor network, as well as its integration into the network using a programmable parametric structure. The proposed approach was developed in SystemC language considering the properties of this language which perfectly fits the needs of both the hardware description of the nodes, as the complex algorithms that can run on them. The proposed model allows to include several physical node features such as data formats, connections between components and memory, and real limitations in runtime, among others. The developed model is applied to a geoelectrical prospection network in order to demonstrate in practice its advantages and application possibilities, and considering that in such kind of networks nodes act as sensors or actuator, depending on the operation mode in which it is working. Two classic network topologies, *chain* and *bus*, are used to test the model with a set of different parameter values. Performance metrics are used to compare the network topologies and their interaction with the node set size, in order to verify the validity of the proposed approach.**

Keywords — Sensor Network, Node modeling, SystemC, Geoelectrical Monitoring System.

I. INTRODUCTION

Measurement of physical parameters is essential for engineers and scientists, because it allows modeling and make predictions about the behavior of a given system or phenomenon. One of the technological approaches most currently used to support this kind of applications are the wireless sensor networks (WSN), which are characterized by employ a big number of interconnected nodes for sensing, processing and communicating collected data to a base station or among the nodes. Because of its advantages, this technology can be used in a wide range of applications such as military, environment, infrastructure, industrial, medical, etc.

WSN is a very popular technology due to the facilities offered by wireless communications and because this allows forming *ad-hoc* networks that characteristically do not require a pre-established physical infrastructure or central administration. Although WSN networks have been extensively studied, there are still some challenges that can affect the design and functioning of the network such as nodes constraints, low-power consumption, overall system cost and communication protocols [1]. For example, nodes in a WSN normally operate under severe restrictions of energy, which means considering simultaneously the distributed signal/data processing, the medium access control and the characteristics of the employed communication protocol [2].

During the design phase, the WSN designers must choose between two options for implementing the system. The first one consist on the use of commercial platforms for wireless sensors such as Imote2 [3] or Mica [4]. In this case, it may be necessary to adapt the chosen platform to meet the specific application requirements, because although the commercial platforms incorporate a number of different configurations, these still can be considered as closed architectures [5]. As a second option, *ad-hoc* sensors can be used to compose the network, taking advantage that these can be carefully designed to suit the target application. However, this option could imply more work for the designer because several hardware and software issues which must be carefully considered for an adequate system development.

Currently, there are a number of software tools suitable for reducing development time and effort in the design of network systems. These tools are mainly focused on characterize the network operation by defining parameters such as the communication protocol, packet size and topology, etc. [6, 7]. However, should be noted that most of these tools are not intended to allow an adequate modeling of nodes at hardware level.

On the other hand, there are several electronic design automation tools (EDA) and programming languages aimed to model, describe and simulate hardware systems. These languages can be used to represent the nodes of a WSN in different levels of abstraction from its high-level (behavior) to low-level (gates or transistors). In some languages, the obtained models can be synthesized for a rapid prototyping and/or final production in hardware platforms as those based on FPGAs, CPLDs or ASICs. However, in many of such languages, it is not easy to achieve a proper modeling of the entire network, mainly because the behavioral aspects of complex systems can be difficult to represent [8].

The basic features of some common programming languages and the abstraction levels that each of them can reach, are depicted in Figure 1. In particular, stand out the features of SystemC language, since it is useful to model systems in a wide range of levels of abstraction, from high to low level, as well as for its resemblance to the traditional C language.

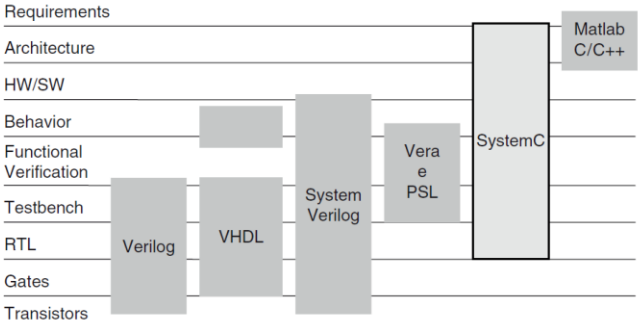

Figure 1. Comparison among different levels of abstraction and the programing languages for systems description. Adapted from [8].

As shown in Figure 1, VHDL and Verilog have a good performance for hardware description, but these languages are poorly suitable to be used for the description of nodes running complex algorithms. Moreover, high-level languages like C++ can be easily used to describe functional behavior of nodes as long as the description of its internal details at hardware level is not required. SystemC language, for its part, can be successfully used for the whole WSN modeling, since it accurately fits the requirements for the hardware description of nodes, the network topology model and to integrate the node into the network. Examples of this characteristic are presented in [9] and [10], in which authors use SystemC for modeling the behavior of nodes and transceivers, along with the features of the network interconnection.

This work presents a programmable structure developed in SystemC aimed to simplify, empower the designing of sensor networks, and fully cover the description of the nodes in the network, both at high (software) and low level (hardware). The proposed approach allows including aspects such as the network topology model, node components, and the integration of the node into the network.

The rest of the paper is organized as follows: Section II presents an approach for the design of a sensor network including hardware and software features of nodes. Section III exemplifies the application of the proposed approach by modeling the nodes for a geoelectrical prospecting application, including for each its algorithmic behavior and internal hardware. Likewise, the sensor network with the programmable structure described in SystemC is verified for the *chain* and *bus* network topologies. Section IV presents the simulation results and the analysis for different sets of sensors in the network. Finally, Section V concludes the paper.

II. Proposed Aproach

This section proposes a set of guidelines to generate the node models based on SystemC, considering both, the software and hardware for nodes, as well as the network topology. The proposed guidelines are aimed to speed up the development time, by facilitate and simplify the design of nodes, and thus, the whole sensors network. In the following, the proposed guidelines are described:

- General Design: The initial stage involves the node specifications, which are defined concurrently with the network topology and the application of the sensors network.
- Detailed Design: Once the system specifications are defined, each node is divided into their internal high-level components. In this stage, the designer specifies the function to be executed by every internal component. It should be noted that some component functionalities are common to all nodes, such as processing, acquisition and communication, while other functionalities are specific and depends on the particular application of each node.
- Node Modeling: In this stage, each component for a node must be modeled at behavioral or detailed level. The selection of the abstraction level of a model depends on the simulation aims. In any case, when the desired outcome is a simulation model, it may be accomplished in a faster and easier way than the one required for synthesis purposes, since is enough with define some features such as dynamic models and parameters (e.g., energy consumption and time delay). Otherwise, when a hardware synthesis is the purpose of the model, it generally requires more work to reach a synthesizable RTL description.
- Network Modeling: In the network modeling, virtual channels are defined to interconnect nodes with each other and/or with a base station (sink). Modeling can be implemented for real or ideal channels, taking into account that for a real channel modeling, it is necessary to include physical features and additional SystemC libraries as SystemC-AMS. The programmable structured compilation of SystemC is a suitable option to instantiate nodes, since it uses loops as the base of the instantiation process. Inside the loop, virtual channels linking the nodes according to the rules described in the network topology.
- Model test: Verification and validation processes are applied to evaluate the correctness and usefulness of the model developed for the network and nodes. Some state-of-art metrics are used to assess the performance of the generated model.

These guidelines may be used in modeling nodes that have an identical functionality, and when must be included specific node parameters such as node addresses (node ID), initial configurations, and patterns for high level performance, among others.

III. An Application Case

In some applications, the design and description of nodes may become critical for the correct network modeling, especially when these must working simultaneously as actuator and sensor. This section provides an example of the use of the guidelines introduced in Section II, for both the design of the nodes as the entire network.

A. *Nodes for Geoelectric DC prospecting*

A type of instrument that employs a set of electrodes as nodes is a DC geoelectrical prospecting equipment, in which, nodes are deployed in a survey area to form a sensor network. In this case, some nodes are used to supply current to the ground (actuator), while others are used to get the voltage from the soil (sensor). From the voltages read on each node, it is possible to identify different ground characteristics, along with the form of the underground structures.

Geoelectrical prospection methods are based on equipotential lines and Ohm's law theories. Electrodes (nodes) can operate in any of four ways: as active current actuator (AA), as passive current actuator (PA), as active voltage sensor (AS), or as passive voltage sensor (PS). Once nodes are configured, two nodes operate as current transmitters supported by an external current supply source, while the remaining nodes operate as voltage receivers. It is worth noting that any node is able to process and send signals to a base station.

The system is designed to allow that each monitoring cycle restarts after each switch operation, operated by an embedded control circuit (MCU) integrated into the node, which initiates a sequence of processes called configuration, acquisition, processing and communication. This circuit is also called a data acquisition unit (DAU) [11]. The Figure 2 represents the general diagram of a DAU.

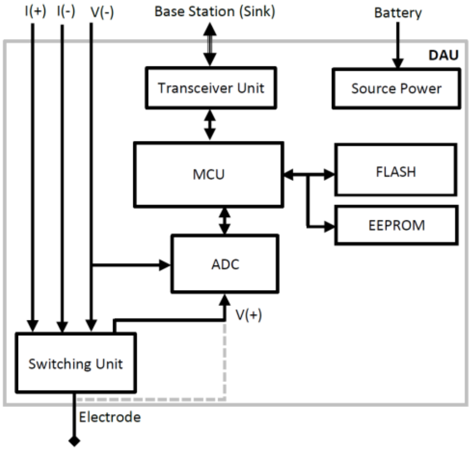

Figure 2. General diagram of internal structure of a DAU. Adapted from [11].

When a DAU is working as a sensor, this allows measuring the node voltage and employ sampling windows to correct external noises. Then, a node which is operating in active mode makes a request of the resulting data from the node working in passive mode, in order to establish the differential voltage between them. The data acquisition process is performed in low frequency (generally in the order of Hz). Next, a node functioning as actuator allows applying a flow of current from an external power source to the ground, through a switching unit disposed inside the node for this purpose.

B. *Node modeling with SystemC and SystemC-AMS*

The design focus of the developed model is on simulation only; therefore, no synthesis activities are performed in this part.

A bottom-up approach strategy is employed to generate the DAU model, as shown in Figure 2. Each node is designed and described independently, and then all of them are joined together to form the system. This model includes physical parameters as time constraints and power consumption.

Table I shows typical parameter values used in the model, which are extracted from datasheets of several off-the-shelf components. These parameters are used to test the node into the system. The developed structure allows replacing critical modules such as communications transceiver and ADCs in order to obtain, and compare, the system operation performance under different conditions.

TABLE I. PARAMETERS FOR DAU MODEL PROVIDED BY THE USER TO SYSTEMC TOOL, DATA ARE EXTRACTED FROM DATASHEETS.

Element / Reference	***Parameter***	
	Time (mS)	***Power Consumption (mW)***
MCU	5	84 / 5*
ADC		
AD7884	0.0053	250
AD7821	0.00066	50
AD7829-1	0.000420	24
TLC5510	0.000125	127.5
TLC5540	0.000025	85
MEMORY		
EEPROM AT28C64B	2	200
FLASH SST39LF010	0.02	18
TRANCEIVER		
CC2520 Tx	1	126.35 / 0.00038*
CC2520 Rx	1	85.88 / 0.00038*
MC13202 Tx	1	102 / 0.00034*
MC13202 Rx	1	125.8 / 0.00034*
XBee Tx	1	148.5 / 0.00033*
Xbee Rx	1	165 / 0.00033*
CC2590 Tx	1	79.56 / 0.000108*
CC2590 Rx	1	12.24 / 0.000108*
MCP2551 Tx	1	375 / 0.050*
MCP2551 Rx	1	375 / 0.050*
SWITCHING UNIT		
CPC1968	4.5	14

* (Power consumption in hibernate mode.)

The model description of the internal node components is divided into two approaches. In the first one, components are modeled by using a behavioral description that focuses on its functionality at software level on the system. In the second one, some components are modeled employing a low-level description which is closer to the hardware behavior.

Much of the communication transceivers and control units are modeled by using a behavioral description, because the complexity of the hardware layer of the communication unit, and the need for modeling the software running on the processor. The software model employs high-level design patterns, thus the internal control unit of the communication transceiver is described by a polling cycle, a monitor of reception, and a couple of register modules connected to transmission module.

The MCU employs a polling cycle and state machines to configure the operation of the ADC, timers, and the switching and transceiver units. State machines work as a sequence of reentrant tasks; therefore, parts of these are executed on each polling cycle.

Memory units are described through buffers functioning as memory cells and a concurrent decoder employed to addressing each of them.

A timer module is composed of a binary up counter, a register to store the limit count value, and a binary comparator for comparing their contents. A flag is set if the counter register reaches the limit value.

The switching unit requires a simultaneous modeling of analog and digital signals since this unit includes a digital decoder and a set of ideal analog switches functioning as solid-state relays, see Figure 3. The SystemC-AMS library provides tools to use different kinds of signals into the modules description.

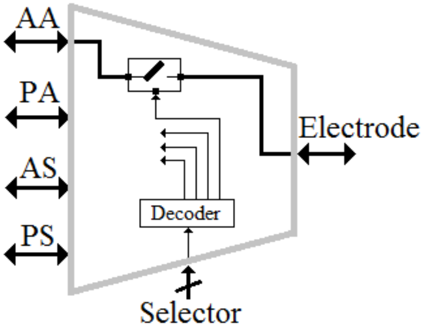

Figure 3. General diagram of the Switching unit as a multiplexer, internal detail of a decoder and switch in the channel between electrode and AA

For the transceivers unit modeling, the input data detection and the connection with the control unit are based on 'events' operating as trigger signals among processes, thus, events are modeled as peripheral interruptions at the hardware level. The MCU identifies the occurrence of an event according to the flag that is set within a status word. In this way; the peripheral modules, connected to MCU, send an event once to trigger and start an internal task in the MCU.

As in the case of the switching module, the ADC modules require the simultaneous modeling of analog and digital modules, which usually cannot be directly modeled by means a discrete style in SystemC. Here again, the SystemC-AMS extension library is used for properly modeling the ADC features. In this example, a simple 'FLASH' ADC architecture is chosen to illustrate the simultaneous modeling of analog and digital modules, see Figure 4.

The 'Sample and Hold' module requires a digital input (Ts) to control the analog input rate to the comparator module. The comparator module is described as a simple analog cell in which the reference value and the input signal values are compared to produce a digital signal.

Each of the comparator cells can be parametrically instantiated to complete the ADC description, taking the desired ADC resolution as the instantiation limit parameter. For simulation purposes, the ADC resolution used in this example is 8 bits. The block for signal conditioning is described as an analog module by using the SystemC-AMS extension library.

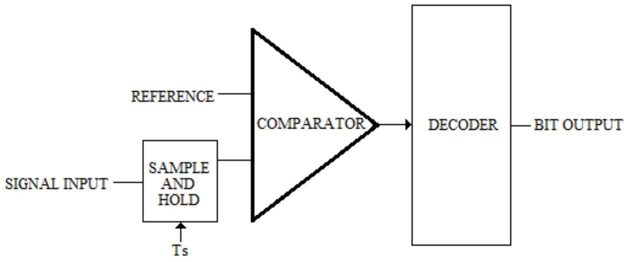

Figure 4. Internal view of a single compartor cell for an ADC FLASH. architecture.

By following the above description methodology, the developed SystemC model includes the parametric values for each component as well as the most relevant restrictions and physical behavior of the system at the hardware level. An exception is the battery, for which a model of energy consumption is not included.

C. *Modelling the sensor network with SystemC*

The network modeling based on SystemC uses structured programming, in which a vector is devoted to storing a set of items composed of nodes. In this case, it is assumed that all nodes are equal making the vector can be easily filled through the nodes instantiation. Dynamic compilation can be exploited for the vector filling taking advantage it allows the simulation of a large number of nodes without the need of major changes in the developed code.

The node parameters and the network interconnections are set through loops designed to simplify the compilation process. Traditional C++ operators as '*for*' and '*while*' are employed to the loops implementation.

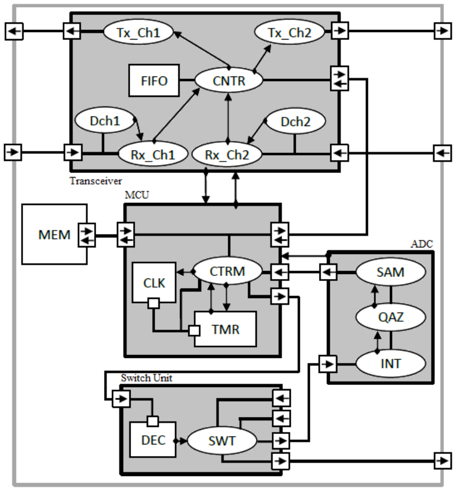

Figure 5. General block diagram of node model with SystemC.

The proposed methodology is also applied to the channel description. At each iteration loop, the nodes are interconnected according to the selected network topology, through channels arranged as vectors. Channels can be modeled as ideal connections based on signals, or may include physical characteristics defined in custom application modules [12]. In this work, the channels are modeled as ideal connections.

Herein, two classic sensor networks topologies are selected to test the structured programming approach: *chain* topology [13], and *bus* topology. The *bus* topology employs the SPIN protocol limited to a cluster. Since the connection of nodes is taken as ideal, channel modeling does not consider the time and energy losses. The main objective of this simulation is to evaluate the performance of nodes under two ideal network topologies, and to compare the energy consumption and time delay; however, the 'loss' and 'error' models may be added to the channel in the programmable network structure.

Figure 6 presents the block diagram for the two topologies, with *n=4*, in which *n* is the maximum number of nodes in the network. For the *chain* topology, each node uses two channel transceivers to the communication with the neighboring nodes, while for the *bus* topology, the transceiver unit only needs one; therefore, the transceiver unit requires minor or null changes in the model description and may be adapted to every network topology. If a channel model is required, it may be easily added between the nodes.

It should be noted that channels in *bus* topology must be a resolved type, i.e., channels are allowed to have multiple data sources, which enables the reception and transmission of data packages from, or to, all nodes in the network.

IV. Simulation

Simulation process starts with the operation of a sink unit. This unit sends a set of data packages to test the nodes functions and the network interconnections.

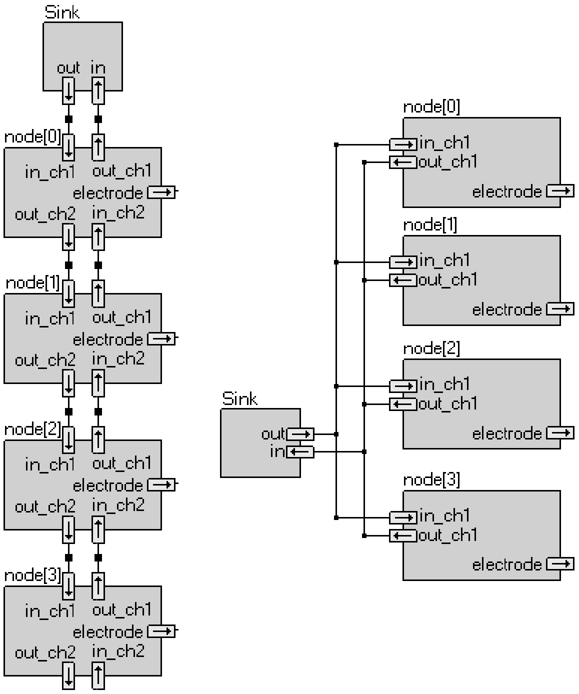

Figure 6. General networks diagram for *chain* (left) and *bus* (right) topologies, with 4 nodes in the network. Diagram generated and modified with extension library from [14].

The sink unit tests the transceiver module by means a set of data packages. When the correct data package arrives to the module, it starts the MCU operation. The MCU configures the internal components and enables the operation of each one according to the information received in the package. Data packages contain commands to enable the components in the node. During simulation process, all nodes receive packages to enabling components on these. In this way, modules as the ADC, switching unit, timer, memories and transceivers are tested for every node, along with their four operation modes, AA, PA, AS and PS.

The previous simulation procedure is applied to various sets of nodes. Every set have different ADC and transceiver settings in order to compare the networks features.

Evaluation metrics estimates the power consumption and delay time for the two topology models evaluated in this work. These metrics employs the nodes density in the network in order to establish a trend.

Two classic metrics are used to evaluate the sensor network. The first one is the 'delay time per package' (DTPP) metric that relates the time delay of the sent data packages from nodes to the sink. The second one is the 'average power consumption per node' metric which establishes the energy consumption of the entire network [15].

In this case, the initial energy of batteries used in nodes for simulation purposes is 4J (parameter assigned once simulation is started). Results for each topology model are obtained with network sizes that range from 2 to 200 units. Delay time is measured for every node, and then, this information is used to establish the DTPP.

Results of DTPP show the trend behavior of the evaluated networks. In the *chain* network, adding nodes to the network affects its performance, making the reception average time of a data packet increases linearly. An indirect packet transmission between nodes nonconsecutive is the factor that more affects DTPP for this topology.

In *bus* network, all nodes have a direct connection with each other and the delay time remains as a constant. As mentioned before, it would be taking into account that in this case, the simulation process evaluates the performance of nodes and network under ideal conditions. Figure 7 presents the results of different node configurations as a function of the average power consumption per node (AECN.)

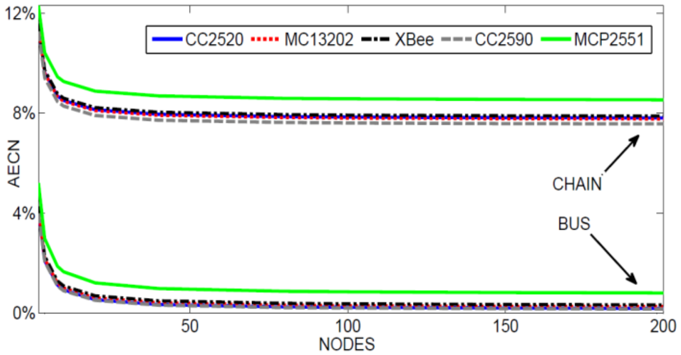

Figure 7. Average energy consumption per node (AECN) for diferent node transceiver parameters in *chain* and *bus* topologies.

According to the obtained results, it can be stated that the *chain* network drains more energy per node than the *bus* network topology. This trend can be explained by the presence of indirect paths between the nodes. It should be noted that the *chain* network topology allows achieving greater distance and network coverage than the *bus* network topology. Moreover, the observed trends in energy consumption show that the *chain* network topology is not useful in sensor networks powered by batteries, whereas the *chain* network topology could be used without any energy restriction.

The AECN results for the different transceiver settings evaluated in this work, show that five of them present a maximum variation of 0.187% in average energy consumption for the *bus* topology. In particular, the MCP2551 has the maximum variation that rises up 0.854%. This results show that transceiver selection process severely affects the total energy consumption in the network. In contrast, a variation on the ADC parameters in the nodes does not represent greater changes in the total energy consumption; this is caused by the limited use of the ADC unit in the nodes in comparison of the transceiver unit.

A second test is done with the two networks. This time, the node includes a hibernation mode to shut down the transceiver and MCU core. Figure 8 presents the results of AECN for different node configurations with hibernation mode.

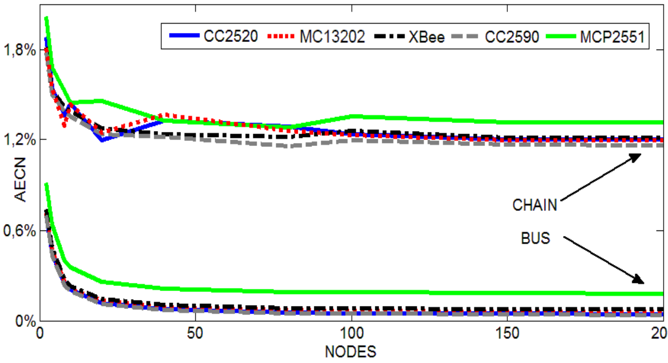

Figure 8. Average energy consumption per node (AECN) for diferent node transceiver parameters in *chain* and *bus* topologies with hibernation mode.

According to results, the *chain* network keeps draining more energy per node than the *bus* network topology, even in sleep mode. In *chain* network topology, the AECN performance it is affected by transceiver settings, in this network, transceivers and cores turn on and off generating the behavior seen in Figure 8 for sets of nodes below 100 units and in some cases, i.e. MCP2515 and MC13202, the AECN is change as a function of the number of elements in the network. After 100 node units the AECN stabilizes to a constant. Moreover, in bus network topology, AECN stability is maintained for all node groups.

With hibernation mode, the energy saved by the networks is close to 6 times the energy consumed in continuous operation mode.

V. Conclusions

This paper presents a set of guidelines for the design of simulation models for nodes in a network model that allows combining high-level design patterns and low-level descriptions into a single network description. Likewise, the developed examples evidence differences and similarities between the two network topologies evaluated.

In the description of sensor networks based on the SystemC language and the SystemC-AMS extension library, the use of the programmable structure, that exploits the facilities of module instantiation and connection, allows to simplify the code implementation and eases performing changes in simulation parameters as energy, delay time and the number of nodes can be added to the network.

References

[1] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *The Journal of supercomputing,* vol. 68, no. 1, pp. 1-48, Apr. 2014.

[2] C. Buratti, A. Conti, D. Dardari *et al.*, "An overview on wireless sensor networks technology and evolution," *Sensors,* vol. 9, no. 9, pp. 6869-6896, 2009.

[3] M. Bauderon, S. Grumbach, D. Gu, X. Qi, W. Qu, K. Suo, and Y. Zhang, "Programming imote networks made easy," In *The Fourth International Conference on Sensor Technologies and Applications.* IEEE Computer Society, pp. 539-544, 2010.

[4] B. Tavli, K. Bicakci, R. Zilan and J. Barcelo-Ordinas, "A survey of visual sensor network platforms," *Multimedia Tools and Applications,* vol. 60, no. 3, pp. 689-726, 2012.

[5] H. Chih-Ming, F. Samie, M. S. Srouji *et al.*, "Hardware/software co-design for a wireless sensor network platform," In *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS),* pp. 1-10, 2014.

[6] X. Xiaodong, S. Weiren, and H. He, "Comparison of OMNET++ and other simulator for WSN simulation," *Industrial Electronics and Applications (ICIEA 2008). 3rd IEEE Conference on*, pp. 1439-1443, 2008.

[7] P. Nayak and P. Shree, "Comparison of Routing Protocols in WSN using NetSim Simulator: LEACH Vs LEACH-C," *International Journal of Computer Applications,* vol. 106, no. 11, pp. 1-6, 2014.

[8] D. C. Black, J. Donovan, B. Bunton *et al.*, *SystemC: From the Ground Up, Second Edition*: Springer US, 2009.

[9] W. Du, F. Mieyeville, and D. Navarro, "Modeling energy consumption of wireless sensor networks by SystemC," In *Proceedings - 5th International Conference on Systems and Networks Communications (ICSNC 2010)*, pp. 94-98, 2010.

[10] M. Vasilevski, N. Beilleau, H. Aboushady, and F. Pecheux, "Efficient and refined modeling of wireless sensor network nodes using SystemC-AMS," in *Conference on Ph.D. Research in Microelectronics and Electronics (PRIME 2008)*. pp. 81-84, 2008.

[11] P. Stummer, H. Maurer, H. Horstmeyer *et al.*, "Optimization of DC resistivity data acquisition: real-time experimental design and a new multielectrode system," *Geoscience and Remote Sensing, IEEE Transactions on,* vol. 40, no. 12, pp. 2727-2735, 2002.

[12] J. Haase, M. Damm, J. Glaser J. Moreno, and C. Grimm, "Systemc-based power simulation of wireless sensor networks," In *Proceedings of the Forum of Design Languages (FDL)*. pp. 1-4, 2009.

[13] A. Abdelhadi, "Identically programmed intelligent electrodes for use in geoelectrical surveys," United States Patent, US7158048-B2, 2007.

[14] J. Turoň, and K. Jelemenská, "Contribution to graphical representation of SystemC structural model simulation," in *Proceedings of the 7th FPGAword Conference,* L. Lindh, V.J. Mooney, S. de Pablo, J. Öberg, Eds. Copenhagen: ACM. pp. 42-48, 2010.

[15] S. Khan, A. S. K. Pathan, and N. A. Alrajeh, *Wireless Sensor Networks: Current Status and Future Trends*. France: CRC Press, Taylor and France Group, 2012.