DESIGNING LOW-POWER WIRELESS SENSOR NETWORKS

Joseph Wenninger and Javier Moreno and Jan Haase and Christoph Grimm

Institute of Computer Technology
Vienna University of Technology
Vienna, Austria
Email: {wenninger|moreno|haase|grimm}@ict.tuwien.ac.at

ABSTRACT

In the ever growing area of wireless sensor networks it is very important to estimate the energy usage of sensor nodes, especially if they should be powered by a battery. Often the accumulated energy consumption of a message delivery across the whole network is of interest too, in order to estimate battery life times, since message forwarding needs energy too. For rapid development cycles, fast but accurate simulations are needed. The approach presented in this paper uses transaction level modeling (TLM) to speedup simulation of wireless communication systems. The fundamental idea is that air communication can be modeled similarly to a multi master bus. Nodes are simulated by employing a multi-threaded instruction set simulator (ISS) and models of the peripheral components. The energy usage for sensing, receiving and transmitting can be estimated in the simulation and this information can then be attached to TLM messages for accumulation related to messages. The example use case throughout this paper is a tyre pressure monitoring system. The modeling and simulation is done in SystemC. [1]

1. INTRODUCTION

Wireless networks become more and more important these days. On the one hand, they start replacing cable bound network infrastructures within buildings, because the effort of laying cables is reduced. There for wireless networks are cheaper in a sense of reduced man hours and less collateral damage in existing (old) buildings, while setting up the network. On the other hand wireless networks, especially wireless sensor networks (WSNs) conquer additional areas of our every day lives. Areas where cable bound communication would not be feasible at all can be connected now. Applications for cargo container tracking, measurement of arbitrary environmental data (eg. CO_2 density, N density, temperature, pressure, moistness and so on) come to mind. Cargo tracking has to cope with the huge distances the tracked items are moved. Environmental data could have the advantage of being collected distributed in public space areas. Another kind of application for WSNs is ”realtime” data acquisition from sensors, being attached to fast revolving surfaces. The example shown throughout this work to explain the various design decisions and the results is an in-care WSN a so called tyre pressure monitoring system (TPMS). Thereby the sensor could be attached either to the valve or to the rim or could even be embedded into the tyre rubber itself. The most challenging factors hereby are that the sensor should work as long as the tyre is mechanically functional. This means the sensor and especially its battery has to cope with big temperature and pressure changes, which influence the lifetime of the battery and thus of the whole sensor system. To enhance robustness and because of technological feasibility the battery is permanently attached to the sensor and molded into the same casing, therefore it cannot be exchanged during the lifetime of the tyre/sensor set. The battery also empties by the value of the current and the time the current has to be provided. Many other things have to be considered too. For evaluation and design purposes concerning life-time, feasibility, reliability, security and robustness simulation is a key factor for time-to-market reduction and system/module cost efficiency. The scenario is shortly explained in Sect. 1.1. The general structure of the simulation model is shown in Sect. 1.2. The detailed explanations of the models and the approach taken to simulate a single node is shown in Sect. 3, whereas Sect. 4 explains the overall network simulation strategies. The paper ends with a conclusion and outlook (Sect. 5).

The described methods and the developed libraries and tools will be a major part of the SYCYPHOS framework, which is currently under development. The SYCYPHOS framework will provide a holistic way to design cyber physical systems (CPS). SYCYPHOS is planned to allow a system level design and then refinements down to circuit level within one framework with support for multi level simulations. In addition to power profiling the framework will also contain components for signal processing and estimations of other quantities, like costs, areas, volumes, etc. . Accuracy and

[1]This work is conducted as part of the Sensor Network Optimization by Power Simulation (SNOPS) project which is funded by the Austrian government via FIT-IT (grant number 815069/13511) within the European ITEA2 project GEODES (grant number 07013).

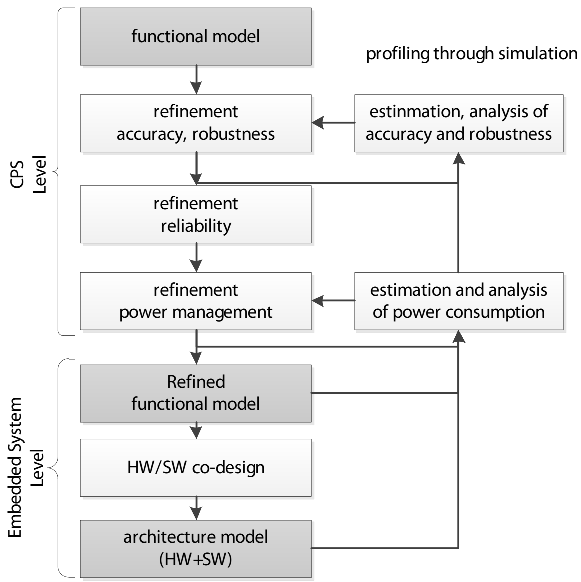

Fig. 1. In this figure the SYCYPHOS design flow of a cyber physical system (CPS) is shown

reliability are another important topic of SYCYPHOS, which will be covered by range based arithmetic to estimate the impact of parameter variations to all types for system performance. The power tracking (power profiling), as described in this paper, attaches to the system level design. Therefore, at least rough, results can be obtained already in first design phase of a top-down design approach. The results will be more and more accurate the more the models are being defined on the way to the final circuit level models, which are used for synthesis in the end. In Fig. 1 the design flow, if the SYCYPHOS framework is used, is shown.

1.1. TPMS scenario

As already shortly stated within the introduction, the example network being used throughout this paper is an in car wireless sensor network. For the ease of explanation a single hop network is being considered only. Each car has four sensor nodes and one central unit or sink node. The simulated environment contains three cars, which perform a take over maneuver. Each sensor node can be woken up by the central unit which queries information about the pressure, after the node has done the measurement and has sent the date back it goes back to sleep mode. The central unit can request retransmits in case of failed data delivery. The node itself measures the pressure also from time to time and if a severe loss is detected it starts transmitting by itself. In the emergency case it sends the message three times to increase the probability to be recognized by the central node. The example scenario is shown in Fig. 2. As can be seen by the circles depicted the passing cars can influence each other mutually, so the simulated scenario also contains noise from external sources.

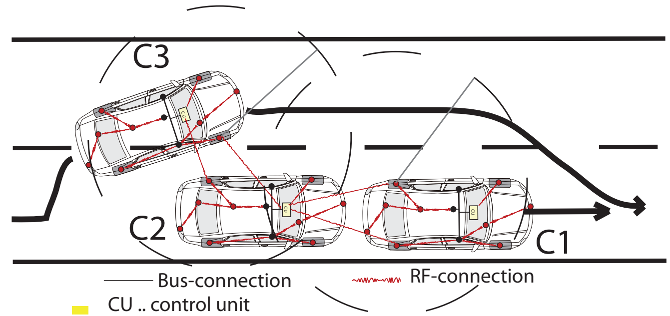

Fig. 2. Simulation scenario, consisting of two cars (C1 and C2) driving in a goose line and a third car C3 doing a take over maneuver. Communications from sensor nodes of nearby vehicles might overlap and thereby conflict, as shown here (see the dashed circles): Car C2's CU receives not only messages from its own tyres but also from C3 and C1.

1.2. Simulation model

The main simulation consists of two levels of complexity. The simple level is the central unit, which is only implemented at a functional level, without any power profiling, since it is considered to be connected by cable to the car battery and therefore being irrelevant regarding energy estimations. The sensor nodes itself are modeled at a higher degree of complexity. One sensor node consists of a capacity based pressure sensor, an micro controller and a transmitter. The micro controller is simulated by usage of an instruction set simulator (ISS) and contains power profiling information. The transceiver simulation is divided into two parts reflecting also the real world implementation. The one part is the low frequency receiver responsible for detecting query calls. This one is simulated in an abstract way, functional, without real influence to power and the high frequency (HF) transmitter part, which is responsible for sending the payload. The HF part is modeled with full power profiling. The pressure sensor is also modeled as a power consuming device. For in-detail information on how the components are connected and their interaction see Sect. 3

The model of the transmission channel is on an abstract level, done with transaction level modeling (TLM), which means the modulation itself is not considered, only general factors like distance vectors, attenuation, noise, signal to noise ratios (SNR) and bit errors, which highly improves simulation speed but still allows the exploration of various scenarios. For in-detail modeling information see Sect. 4.

2. RELATED WORK

Works tightly related to the topics considered within this paper are SystemC, which can be found at [1] and the transaction level modeling (TLM) approach based on this core framework, the OSCI TLM2.0 specification, which are explained in [2–4]. Another work needed to understand the foundation of TLM based simulation is found within an application node for creating a loosely timed system on chip (SoC) model with TLM 2.0, which can be found in [5].

Related to the network level simulation, another approach of doing a WSN simulation is taken in [6], which is not based on TLM, but on the analog/mixed signal extension for SystemC. Other relevant frameworks that might not be left out if it is about simulation of networks, although they are not SystemC based, are NS-2, OMNet++ and PAWiS. NS-2 [7] is a complete network simulator on its own, that can be scripted via TCL (tool command language). Another network simulator is OMNeT++ [8], which has been used in the PAWiS [9, 10] project. Related to the concepts used in the this paper, PAWiS is the ancestor. Informations about the OMNeT++ based air object/communication can be found in [9, 10].

[11] uses a finite automaton to do power aware simulation of a sensor node. Another work, spanning the gap from network to node level simulation is [12].

Works related to instruction set simulation are shown in [13–17]. [13] simulates an ARM7 and a SPARC microprocessor. Some parts are statically compiled, others are dynamically handled.

3. IN-NODE SIMULATION AND POWER TRACKING

The core of the in node simulation is the instruction set simulator (ISS). Three versions have been developed during the project time, two have been developed by students as part of their master theses together with one of the project partners. Those two simulators are for the intelligent state machine (iSM) by Infineon, one is precompiling, the other one is an interpreting simulator. Based on that work a precompiling ISS for the ATtiny88 microcontroller from Atmel has been created [18]. The Atmel controller has the advantage, that it is publicly available and is used within automotive designs. The ISS is implemented in plain C++ and is wrapped with sensor node specific bindings for the used proprietary - SystemC based - simulation framework, or for connectivity with plain SystemC. The second approach has the advantage that only two time lines have to be synchronized, instead of three (ISS,framework,SystemC). Additionally to the synchronization of time lines, the bindings help to determine the state of the node (receiving, sensing, generally running, building packets, transmitting and sleeping, or additionally retransmission and critical pressure status transmissions). The power consumption of the application code execution is accumulated by counting the instruction cycles and logging the average current needed while execution of each cycle. This way a value set of current over time is generated, showing the difference between sleeping and working time. During sleeping periods of the controller a little bit of variants of the standby current is considered. The ISS of each node runs in

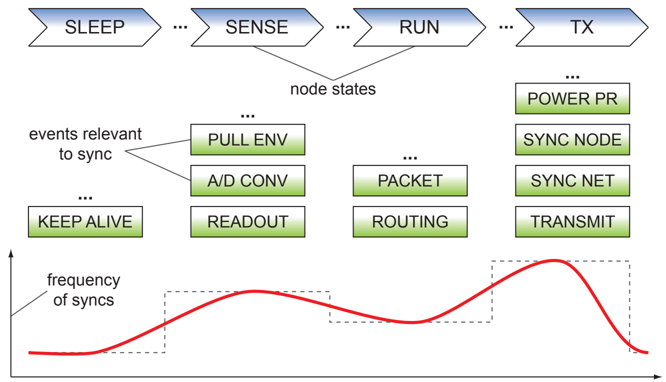

Fig. 3. Adaptive time and energy usage synchronization based on the node state

an own thread, to allow decoupling from the main simulation, this way a speed up of the simulation is achieved, since it can run in parallel until a synchronization point is reached. Synchronization points are created every hundred (or even more) clock cycles during sleep periods of the node or at each basic block boundary during running times. The instruction count between synchronizations is arbitrarily chosen, in that way, that during sleeping periods the standby current does not vary much, so the whole energy during sleeping can be estimated by a simple product. The window must not be too large though, because the larger the interval during sleeps the larger are the delays of incoming external interrupts. It is not a big problem to slightly delay the interrupts during sleeping periods, but it increases simulation performance a lot, since nodes are sleeping most of the time anyway. The larger synchronization window during sleeping periods reduces the number of thread synchronizations, which speeds up the overall simulation. The trade-off between simulation performance and accuracy is valid, since for long simulation runs the slight delays can be neglected. Figure 3 shows the simulation behavior. While sensing the tyre pressure additional to the micro controller an increased energy consumption within the sensor part adds to the node's power consumption. After the transmission package has been created the transmitter adds to the energy usage. This energy consumption is tracked together with the location of the node. This is especially needed for visualizing the scenario and is needed for the overall network simulation, too, see Sect. 4. All components building together a node, are shown in Fig. 4. The graphical user interface for the visualization is explained in [19].

The multi threaded approach, in addition to the precompilation, allowed a simulation of up to 140 nodes (i7 Quad-Core 860 at 2.80GHz), before reaching real time simulation speed. This means, that the simulation for the chosen scenario always runs much faster than real time, since this scenario contains only a few nodes. For instance three simulated cars have a count of 15 nodes (12 sensor nodes and 3 control

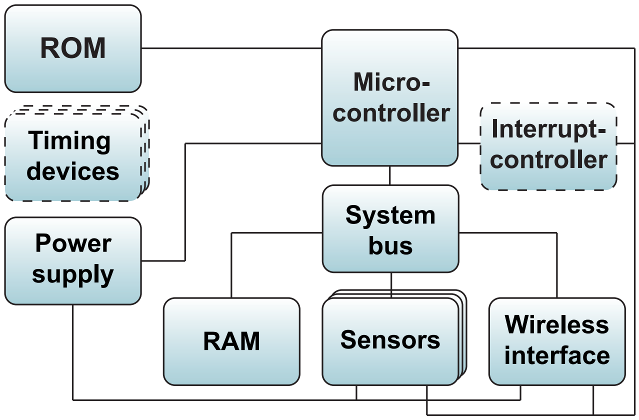

Fig. 4. This figure shows all (simulated) components of a node. In the described simulation the energy consumption of the RAM and ROM are accounted to the instruction execution within the ISS. Aging of the battery is only considered in a limited way. As the maximum of the energy it can provide the value is taken, that represents the specified maximum energy after ten years of shelf time. The nodes accumulated power consumption may not reach that limit earlier.

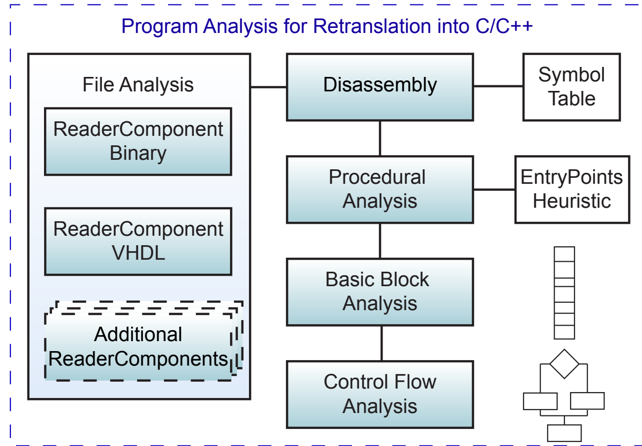

Fig. 5. This image shows the steps performed while preparing the machine code of the simulated target architecture for recompilation to native simulator code.

unit (sink) nodes) The simulation time increases somewhat exponentially with the number of nodes, since many nodes mean much more synchronization effort and much more disk thrashing due to writing of the log data. Precompiling ISS means, that the machine code for the AVR controller (which has been compiled with the AVR studio) is taken, analyzed and parsed. After parsing of the machine code the machine instructions are translated into C++ macros and function calls, with addition of synchronization primitives at basic block boundaries for the simulation. The steps performed can be seen in Fig. 5. This C++ code is natively compiled for the machine running the simulation, in this case for x86 or x86_64 architectures. This re- or precompilation is done to allow the evaluation of the real system firmware code on the one hand and to allow fast simulation runs on the oder hand, since code native to machine which runs the simulation is executed much faster than interpreted foreign machine code. This native code is then linked either dynamically or statically to the executable specification. This increases the speed of the simulation immensely. If jumps to addresses occur, which can not be determined during analyzing, a switch to the interpreting simulator is done. If the interpreter reaches an address again during execution, which is well known, a switch back to the precompiled version is done, to get the higher speed gain again.

4. NETWORK SIMULATION AND POWER TRACKING

The base for the chosen approach is the fundamental idea in the similarity of communication channels. A bus system connects various address and data lines of different modules over a shared medium, the bus. For WSN nodes the abstract air, which is given by the environment as a communication channel, connects the various transceivers to each other. The communication over the air can be abstracted, with some limitations, into the communication pattern of a serial bus. This analogy is shown in Fig. 6. Things that have to be considered over a ”normal” bus, are that a message from a source A to a sink Z does not have to be a one-hop direct communication. Depending on the geometry of the system, a message might have to be relayed by intermediate nodes. With multi-hop routing each node receiving the message, because it is within reach, may forward the message, therefor a message duplication could occur. The destination node or intermediate nodes might receive the initial message either directly or from one or more intermediate nodes. The other thing to consider is, that a given node might be able to receive a given message, because the transmission’s energy level is high enough, but noise, generated by other concurrent messages within the environment or other unrelated transmitters might inhibit successful reception.

4.1. Communication and tracing model

The problem of multi-hop or duplicated messages (one message that takes more than one path in parallel) is solved by introducing split transactions. The power consumption of all the split parts are accumulated, since every transaction ”branch” adds to the overall power consumption within the system for the transmission from A to Z.

The fact that some nodes might not receive messages due to noise (in the real world), is modeled by adding bit error rates at the position nodes, based on the power of the signal to be received and the powers of other messages being con-

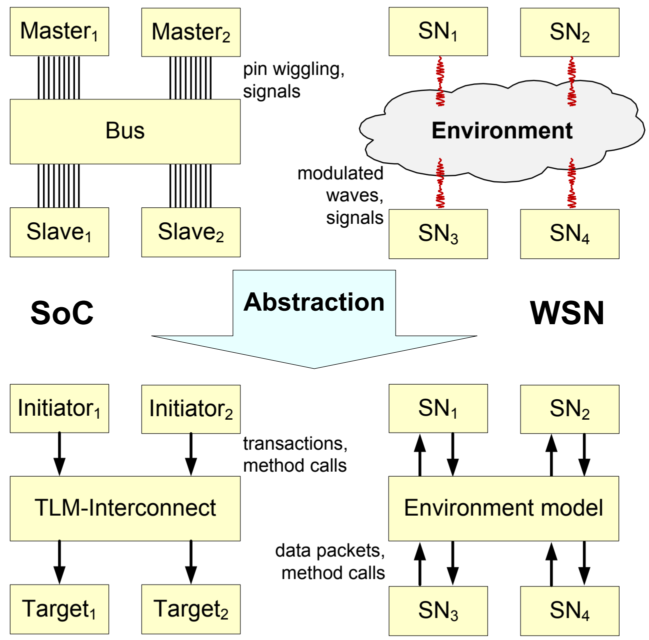

Fig. 6. Similarity between TLM based memory mapped system buses and air communication within wireless sensor networks.

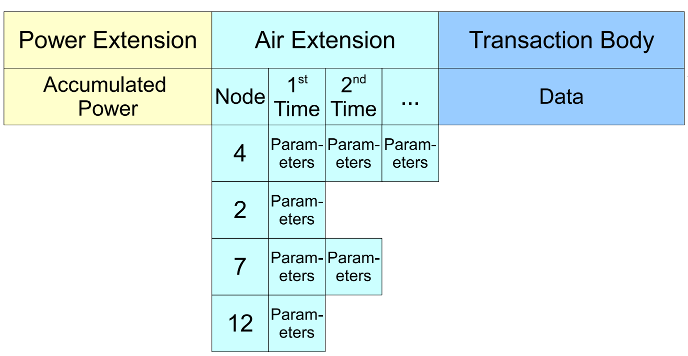

Fig. 7. Concept how energy and routing information is attached via TLM generic payload extensions to the simulated transmission payload

currently transmitted through the air. It is some high level signal to noise ratio (SNR), since no signal wave forms are considered but just the average power levels based on packets. The transmission is modeled as packet based information exchange.

The accumulated power consumption and the information about transaction splitting and the transmission / noise power levels are attached to the simulated payload via TLM framework generic payload extensions. Figure 7 shows the concept of the information attachment. As a conclusion of this part, the overall message, until it reaches the end point or gets lost within the air model/interconnect is accounted within one transaction. The amount of power, which is accumulated at each step, is retrieved from the energy consumption collection of the single nodes. Some nodes might by modeled at an abstract level, for instance always using a fixed amount of energy, while most of the nodes are being simulated at full detail level as described in Sect. 3. More and different aspects on this topic are explained in [20, 21]. In [19, 22] it is shown how to integrate the developed abstraction with proprietary simulation environments.

5. CONCLUSION AND OUTLOOK

The approach taken has proven to be very efficient concerning simulation time and ease of use. It is planned to use the SystemC only framework within future projects and there are already research projects which are using the intermediate results of the framework creation. Features which would be interesting to implement in the future would be the integration with SystemC AMS to easily track power consumption of analog and mixed signal parts. Production technologies could have a large impact on those analog parts, therefor technology data has to be incorporated to lead to meaningful results. Adding this while keeping high level simulation and high simulation speed are still a big challenge to solve. The biggest challenge will be to provide a common, easy to use API for adding power profiling and estimation features to all design levels and integration with range based arithmetic to be able to estimate how parameter tolerances have an impact on the power consumption. Finally, SYCYPHOS will provide all of that.

6. REFERENCES

[1] Open SystemC Initiative, *SystemC™*, http://www.systemc.org.

[2] Open SystemC Initiative, *OSCI TLM2.0*, http://www.systemc.org.

[3] John Aynsley, "OSCI TLM2 User Manual," Tech. Rep., Open SystemC Initiative, 2008.

[4] John Aynsley, "OSCI TLM-2.0 Language Reference Manual," Tech. Rep., Open SystemC Initiative, 2009.

[5] J. Bennett, "Building a Loosely Timed SoC Model with OSCI TLM 2.0," june 2008, Embecosm Application Note 1, Issue 1.

[6] Michel Vasilevski, Francois Pecheux, Nicolas Beilleau, Hassan Aboushady, and Karsten Einwich, "Modeling and refining heterogeneous systems with systemc-ams: Application to WSN," *Design, Automation and Test in Europe Conference and Exhibition*, vol. 0, pp. 134–139, 2008.

[7] S. McCanne and S. Floyd, "NS Network Simulator - version 2," Website, 1996, http://www.isi.edu/nsnam/ns.

[8] Andras Varga, *OMNeT++ Discrete Event Simulation Sytem User Manual*, 29. Mar. 2005.

[9] D. Weber, J. Glaser, and S. Mahlknecht, "Discrete event simulation framework for power aware wireless sensor networks," June 2007, vol. 1, pp. 335–340.

[10] PAWiS, "PAWiS Simulation Framework," 2009, http://pawis.sourceforge.net/.

[11] Animesh R. Tayal, N. V. Choudhary, and Madhuri A. Tayal, "Simulation of sensor nodes for energy conception in wireless sensor networks using finite automata," in *ICAC3 '09: Proceedings of the International Conference on Advances in Computing, Communication and Control*, New York, NY, USA, 2009, pp. 685–688, ACM.

[12] Johann Glaser, Daniel Weber, Sajjad A. Madani, and Stefan Mahlknecht, "Power aware simulation framework for wireless sensor networks and nodes," *EURASIP Journal Embedded Syst.*, vol. 2008, pp. 1–16, 2008.

[13] Mehrdad Reshadi, Prabhat Mishra, and Nikil Dutt, "Hybrid-compiled simulation: An efficient technique for instruction-set architecture simulation," *ACM Trans. Embed. Comput. Syst.*, vol. 8, no. 3, pp. 1–27, 2009.

[14] Todd Austin, Eric Larson, and Dan Ernst, "Simplescalar: An infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, 2002.

[15] Emmett Witchel and Mendel Rosenblum, "Embra: fast and flexible machine simulation," in *SIGMETRICS '96: Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, New York, NY, USA, 1996, pp. 68–79, ACM.

[16] Bob Cmelik and David Keppel, "Shade: a fast instruction-set simulator for execution profiling," in *SIGMETRICS '94: Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, New York, NY, USA, 1994, pp. 128–137, ACM.

[17] Stefan Pees, Andreas Hoffmann, and Heinrich Meyr, "Retargeting of compiled simulators for digital signal processors using a machine description language," in *DATE '00: Proceedings of the conference on Design, automation and test in Europe*, New York, NY, USA, 2000, pp. 669–673, ACM.

[18] Jan Haase, Mario Lang, and Christoph Grimm, "Mixed-Level Simulation of Wireless Sensor Networks," in *Proceedings of Forum of Design Languages (FDL) 2010*, 2010.

[19] Joseph Wenninger, Javier Moreno, and Jan Haase, "Power optimization of wireless sensor networks at design time," in *proceedings of IEEE AFRICON 2011, Zambia, Livingstone, to be published*, 2011.

[20] Javier Moreno, Jan Haase, and Christoph Grimm, "Energy Consumption Estimation and Profiling in Wireless Sensor Networks," in *ARCS'10 Workshop Proceedings*, 2010, pp. 259 – 264.

[21] 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)*, 2009.

[22] Joseph Wenninger, Markus Damm, Javier Moreno, Jan Haase, and Christoph Grimm, "Multilevel Sensor Node Simulation within a TLM-like Network Simulation Framework," in *ARCS'10 Workshop Proceedings*, 2010, pp. 211 – 216.