Development of the MAC layer basic functionality in an IEEE 802.15.4 standard based WSN Simulator

Javier Navarro Espino, Roberto Esper-Chaín Falcón, Félix B. Tobajas Guerrero
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria (ULPGC)
Las Palmas de Gran Canaria, Spain
jnavarro@iuma.ulpgc.es, esper@iuma.ulpgc.es, tobajas@iuma.ulpgc.es

***Abstract*— Simulators play a key role for almost any engineering-related task. Specifically, network simulators are used to depict and evaluate communication protocols. The goal of this work is to develop the MAC layer basic functionality within a major global project, which aims to build a WSN Simulator using Verilog HDL, and to check out both the MAC layer and its integration with the PHY layer, previously developed.**

Keywords— Simulators, Wireless Sensor Networks (WSN), communication, protocols, Medium Access Control (MAC), Physical (PHY), Verilog Hardware Description Language (HDL).

I. INTRODUCTION

Nowadays, Wireless Sensor Networks are grabbing the researchers' attention because they have great potential to develop smart and useful projects. In these networks, sensors are usually embedded in the environment and for this reason, it is a big challenge to guarantee a successful communication between them [1]. Due to the nature of the WSN, simulators reveal themselves as critical tools to study and design the networks before their set up. Simulators allow researchers to make more effective, efficient and less expensive sensor deployments.

Main simulation tools like OMNeT++ or Network Simulator, do not have a full-implemented model of the IEEE 802.15.4 standard, which is essential for WSN. From this point, the project task group thought about the use of HDLs to avoid some of the weak points shown by OMNeT++ and Network Simulator [2]. Actually, this decision brings some advantages like: ease to describe behaviors, highly concurrency, use of checked and stable tools, or possibility to describe bit-level detailed operations.

This paper presents the design and development of the modules that conform the MAC layer of the IEEE 802.15.4 standard [3]. This implementation tries to be as modular as possible, distributing the responsibilities of the whole layer between different sub-modules, managed by a smart-central module. The top module has to be consistent with the previous PHY layer development to ease a successful integration among layers. As a result, different tests within a verification process have been created to check all modules.

II. DESIGN AND DEVELOPMENT

Five modules have been developed, named *mac*, *mac_cpu*, *get_phy_req*, *arbiter* and *bd*. For this work, in order to cover the basic functionality of the MAC layer, have been implemented three service primitives, *MLME_GET* (it can manage MAC and PHY PIB Attributes), *MLME_SET* (it can only manage MAC PIB Attributes) and *MLME_RESET*.

A. MAC Module

This module is the top of the architecture and describes the MAC layer. Its functions are to manage all the access to the physical channel and to provide an interface between the Service Specific Convergence Sublayer (SSCS) and the PHY layer. The I/O interface and its internal modules are represented in Figure I.

B. MAC_CPU Module

This module has the responsibility for managing all the MAC layer activity. It controls input and output data queues, command parsing and processes or passes PIB (*PAN Information Base*) parameters depending on their MAC or PHY nature.

C. GET_PHY_REQ Module

This module is specific for the MLME_GET primitive, because it needs to communicate with the PHY layer. There will be one module for each primitive that needs to send and receive information from the physical layer. In this case, the module is conformed by two Finite State Machines (FSMs).

D. ARBITER Module

This module grants access to the PHY layer for those modules that make a request. The *arbiter* comes up as a solution for the potential problem of different modules trying to access the same resource (PHY) at the same time.

E. BD Module

To complete the control access system developed to communicate properly MAC internal modules with the PHY layer, a communication bus is required, so each MAC internal module that implements a primitive will have a *bus driver* module connected to their outputs. This driver consist mainly of one FSM that manages the input data and send it octet by octet with the output data request and a synchronization clock.

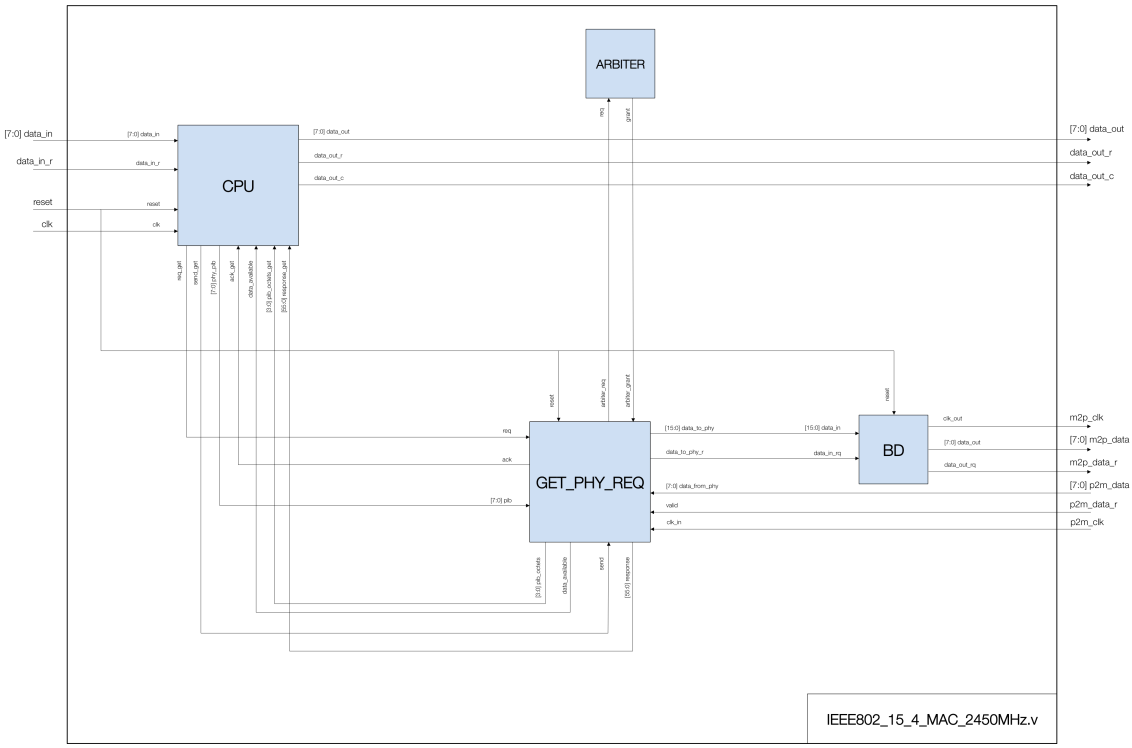

Figure I - MAC Module architecture

III. FUNCTIONAL VERIFICATION

All modules developed have been properly verified using different testbenches for each one and for their integration.

The final verification of the MAC layer consists of sending service primitives and PIB Attributes (except from *MLME_RESET*). The top module have to process the request, to communicate with the PHY layer if needed, wait for the PHY response, and send the data to the higher layer using its output data port.

In Figure II, the correct operation of the *MLME_GET_request* primitive with a PHY PIB Attribute (*phyCurrentChannel*) is represented. As it is depicted, the higher layer issues the primitive and requests a specific PIB Attribute. This is coded as 0x0D | 0x00. After this, the MAC layer issue the *PLME_GET_request* to the PHY, including the PIB Attribute, all coded as 0x0C | 0x00.

The next signals show the response from the PHY layer. This response consists of the primitive *PLME_GET_confirm*, a *SUCCESS* state, the PIB identifier, the PIB value (channel 11) and the Separator octet. The codification for this message is: 0x0D | 0x07 | 0x00 | 0x0B | 0x00. At last, the MAC layer have to issue its output to the higher layer. To make this possible, first, it have to translate the PHY state to a MAC state. For this reason, 0x07 codification is transformed to 0x00, which is a MAC *SUCCESS*. The output message is conformed by a *MLME_GET_confirm* primitive, a *SUCCESS* state, the PIB identifier, the PIB value and the Separator. All this is coded as 0x0F | 0x00 | 0x00 | 0x0B | 0x00.

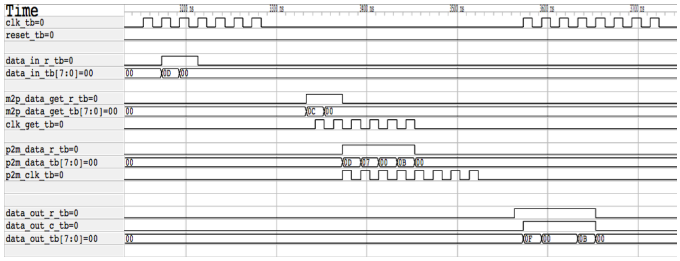

Figure II - MAC + PHY Integration Verification

IV. CONCLUSIONS

The MAC layer basic functionality has been developed creating five modules, which manage three services primitives, *MLME_GET*, *MLME_SET* and *MLME_RESET*. In this work, the previous PHY layer implementation and the IEEE 802.15.4 standard have been deeply reviewed to be consistent with them.

This implementation has been verified, ensuring its behavior for the purpose of continue its future development, completing the MAC layer functionality for the IEEE 802.15.4 standard.

REFERENCES

[1] C.S. Raghavendra, K.M. Sivalingam, T. Znati, Wireless Sensor Networks, United States of America: Kluwer Academic Publisher, 2004.

[2] K. Wehrle, M. Günes, J. Gross, Modeling and Tools for Network Simulation, Germany, Springer, 2010.

[3] Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE 802.15.4-2006.