Architecture and Design of a Bluetooth Low Energy Controller

Pawel Wiecha, Marek Cieplucha, Patryk Kloczko, Witold A. Pleskacz
Institute of Microelectronics & Optoelectronics
Warsaw University of Technology
ul. Koszykowa 75, 00–662 Warsaw, Poland
e-mail: {p.wiecha, p.kloczko}@stud.elka.pw.edu.pl, {m.cieplucha, w.pleskacz}@imio.pw.edu.pl

***Abstract*—This paper describes the architecture and design of a Bluetooth Low Energy Controller. The designed controller consists of a hardware and software part.**

The hardware part consists of a link layer and a physical layer (analog radio), which is out of the scope of this work. Great emphasis was placed on minimizing hardware area and power consumption. Most of the complex and sequential functions were moved to the software Intermediate Layer introduced in this paper. Furthermore, Single Port RAM (SPRAM) was used to reduce the number of registers in the design. Flexibility was achieved by division of the hardware into two separate clock domains. Hardware part was designed using fully synthesizable Verilog HDL to simplify integration process into a System on Chip (SoC).

In the proposed architecture the standard Bluetooth software stack may be used for the controller management. The controller architecture, implementation and verification strategies are described in this paper.

***Keywords*—Bluetooth Low Energy, Link Layer Controller, Verilog Design, Hardware Verification, Intelectual Property**

I. Introduction

Bluetooth specification [1] defines the Bluetooth Host and the Bluetooth Controller. The connection between these two parts is defined using the Host Controller Interface (HCI) which is a set of services and events. Basically, Bluetooth Controller is an implementation of these services provided for the Bluetooth Host. The Buetooth Host is a software protocol stack using a transport layer for HCI services execution. Several open-source Bluetooth Host stacks are available e.g. [2] [3]. Bluetooth Low Energy (BLE) is a simplified Bluetooth standard suitable for low-power applications.

BLE Controller specification defines a link layer and a physical layer. This means that a full controller design requires an implementation of HCI services by a link layer and a hardware-based radio module. However, it is not strictly defined how these layers may be divided between hardware and software parts. This may depend on the application requirements.

There is a number of related publications discussing Bluetooth implementation. Bluetooth Baseband Module implementation for a System-on-a-Chip (SoC) is presented in [4]. Some functions of this module were implemented in software running on an embedded microcontroller. Low-level, hardware-efficient functions were implemented as hardware blocks written in Verilog HDL.

Bluetooth Low Energy Technology has a number of applications. A medical application (Mobile Electrocardiogram Monitoring System) is presented in [5]. In this work the Bluetooth module is responsible for sending the ECG data from signal acquisition module to the analysis platform (e.g. a smartphone). That paper claims that using BLE protocol significantly reduces system power consumption compared to traditional wireless medical devices.

The scope of this work is the design of a BLE link layer defined as a HW/SW-based implementation of HCI services and its verification, with a simplified physical layer simulation module.

II. Architecture

In this section the architecture of the designed controller is presented and shown in Fig. 1. This controller is a part of a System-on-a-Chip (SoC) consisting of several components, including a CPU. One of its main tasks is a real-time communication using BLE protocol. Consequently, Bluetooth Host and Bluetooth Controller are supposed to be integrated in a single device. In this way a certain implementation flexibility is obtained and some link layer operations may be performed by the CPU.

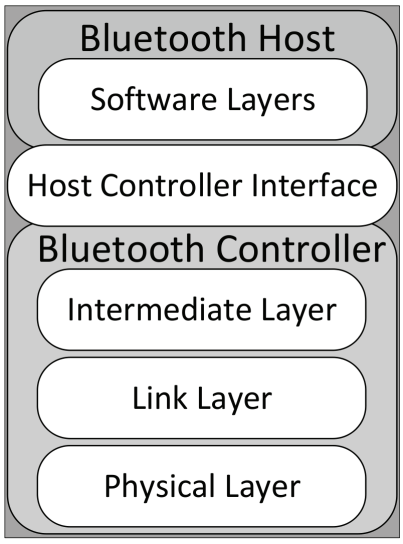

Fig. 1. Bluetooth System.

ISBN 978-83-63578-08-4

The Physical Layer (PHY) is the analog part of the controller. It contains devices responsible for analog modulation and translating digital symbols. It is controlled by the Link Layer controller (the term Link Layer refers to the digital hardware part of the designed controller, instead of the specification-defined link layer). The task of the Link-PHY connection interface is to transmit data bits and select a physical channel.

The Intermediate Layer, introduced additionally, implements some controller functions using the software stack. The Intermediate Layer implements HCI services in the software and translates them into basic register/data operations of the Link Layer controller. The CPU communicates with the hardware over the internal on-chip bus APB [6]. The interrupt mechanism is used to report events and command completion to the CPU. This approach significantly reduces the complexity of the hardware and maintains a clear division between Bluetooth Host and Bluetooth Controller with HCI. Additionally, the HCI interface from the Host perspective remains unchanged.

The Intermediate Layer is implemented in C and is responsible for:

- passing HCI method arguments to appropriate Link Layer controller registers,
- data buffer operations,
- argument validation,
- managing the controller internal timers,
- reporting controller events to HCI.

The main tasks of the hardware part are:

- bit stream processing (CRC calculation, data whitening),
- Protocol Data Units (PDU) framing and analysis (CRC correctness, device address filtering, access code generation and detection),
- managing the BLE Link Layer state,
- controlling protocol timing,
- executing Host commands and reporting results,
- calculating frequency hopping parameters,
- detecting Bluetooth devices.

III. Implementation

The detailed structure of the Link Layer is shown in Fig. 2. There are 2 interfaces: system interface using the APB bus and physical layer interface – connection to the analog PHY. Considering that, the functionality of this layer has been divided into two clock domains. System domain is the CPU clock domain and the clock frequency may be arbitrary, depending on other system requirements. PHY requires separate clock domain, as the symbol timing intervals are fixed by the radio specification. Symbols are transmitted between Link and a PHY using 1-bit wires synchronously with clock generated in PHY. Synchronization submodule is used for synchronizing both data and control signals between both clock domains. A great effort was made to address Clock Domain Crossing (CDC) issues. Various methods, described in [7], were tested and implemented to solve these issues.

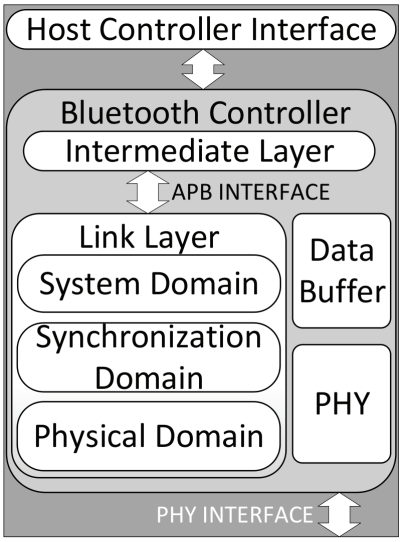

Fig. 2. Bluetooth Controller.

A detailed block diagram of the Link Layer is shown in Fig. 3. All synthesizable modules are encapsulated in core level. Chip level (used in simulations) consist of the core, a Single Port RAM model (a data buffer) and a behavioral, simplified PHY model. SPRAM was used to decrease chip area. Single Port was chosen because there is no need for simultaneous read and write to the memory. Main modules of the designed controller are listed below. System domain contains:

- Bus Controller, which is connected with CPU using the ARM AMBA APB interface and is responsible for read/write control registers and memory buffer,
- CRS (Controller Register Set), which contains control/status registers used to configure the controller,
- RAM Controller, which manages the external memory operations,
- Whitelist Controller, which compares received device address with a list of acceptable devices stored in RAM,
- Connection Manager which implements Link Layer states of operation (e.g. advertising, scanning) using FSM (finite state machine). It also manages functional timing requirements and manages other modules.

Synchronization domain consists of:

- Control signals synchronization registers,
- Asynchronous FIFO, used to synchronize PDU payload between domains.

PHY domain contains:

- Serialiser/Deserialiser (SerDes), which controls data flow between FIFO and Bit Processing Unit,
- Bit Processing Unit, responsible for CRC generation/checking and data whitening (scrambling),
- Packet Manager, which manages outgoing PDU framing and incoming PDU analysis – it appends/decodes various PDU fields such as: preamble, access address, header and the device address,

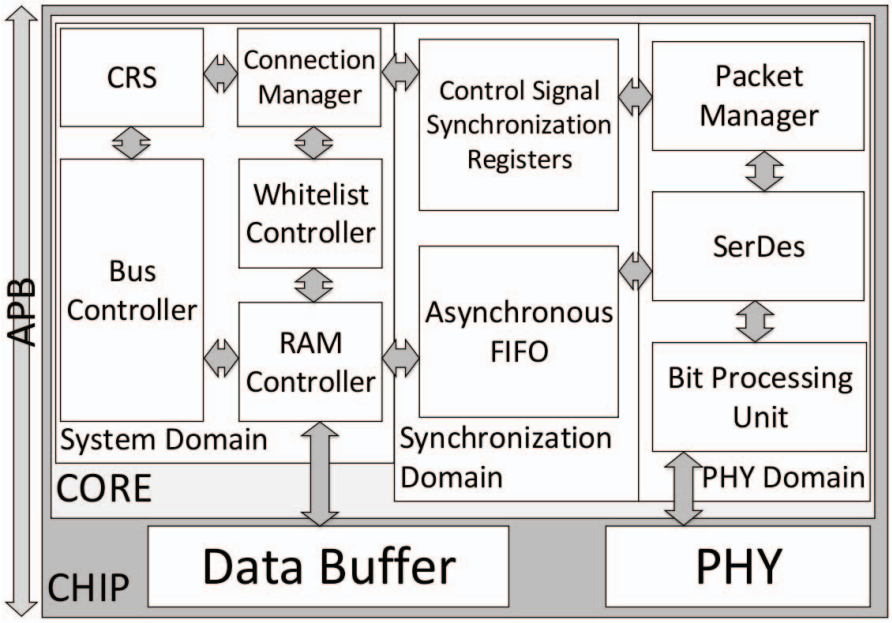

Fig. 3. Link Layer block diagram.

- PHY, a Bluetooth radio – a functional model implemented in SystemVerilog used to translate digital data stream to analog transmission, consisting of the transmitter, receiver and switch.

Data flow of implemented Bluetooth Controller is as follows:

- *Sending data*
  - Host configures the Controller using APB interface,
  - Connection Manager changes Link Layer state and activates Packet Manager,
  - Packet Manager assembles packet header, sends it to the PHY, activates SerDes and RAM Controller,
  - RAM Controller retrieves data from memory and sends it to the SerDes,
  - SerDes deserialises data and sends it to Bit Processing Unit where data is scrambled and CRC is added,
  - PHY transmits data using the analog interface.
- *Receiving data*
  - Packet Manager retrieves PDU header from data stream, activates the Bit Processing Unit, SerDes and RAM Controller,
  - Data is descrambled and CRC is checked. SerDes deserialises data into 32-bit slices and sends them to the RAM Controller,
  - Packet Manager informs Connection Manager about received packet and its type, Connection Manager enables Whitelist Controller,
  - Whitelist Controller checks if the received device address is present on the whitelist,
  - Connection Manager saves received packet data in CRS and RAM and informs upper layer by an interrupt, Host reads event code and data from CRS and RAM.

IV. Verification

The following strategies are used for the controller verification:

- submodule-based testing,
- direct back-to-back testing,
- hardware/software co-verification,
- constrained random verification of the full controller.

In submodule-based testing each block was individually verified in simulations using dedicated standalone testbenches implemented in SystemVerilog. Each feature of the designed block was tested to ensure that unexpected bugs have not been introduced into the design. Specific stimuli were applied to the module to compare the collected output responses with the expected values conformant with design specification. There were 3 types of tests used:

- directed tests: each module was checked with directed test scenario and constant data,
- directed tests with random data: random input vectors were applied in directed tests,
- tests with random error injection: errors are injected to data to ensure that DUT (Design Under Test) correctly handles incorrect data, which verifies error detection, reporting, and recovery features.

As directed tests are not able to verify the internal logic of the module, System Verilog Assertions are used to help find bugs faster. Assertions were introduced at an early stage of the work to support the design process.

Directed back-to-back tests were introduced after all modules have been initially implemented and integrated. Most of

them consist of a simple data exchange between two devices in corresponding modes. As mentioned in Sec. III, the Controller interacts with other devices using the PHY model. As shown in Fig. 4, PHY models of 2 devices are connected together using the physical channel, which is a simplified radio air interface model.

A typical test consists of 2 instances of the Chip communicating with each other in different modes. Each chip may be configured as an advertiser, scanner or initiator. Both devices are configured using the APB interface. This approach detects any functional errors related to data integrity and device-to-device communication. A typical test scenario is as follows:

- both devices are reset and initially configured,
- first device is configured in advertising mode,
- second device is configured in scanning mode,
- devices are exchanging data on the channel,
- data collected from both devices is compared and checked.

After the implementation of the Intermediate Layer, back-to-back testing that required a manual CRS setting could be replaced by the implemented software. The test scenarios are written in C and the test harness is reused from back-to-back testbench. The communication between SystemVerilog testbench top level and C-based software is possible using the SystemVerilog direct programming interface (DPI) mechanism. Tests are designed as HCI command sequences, so the scenario consists only of a few command executions on both devices. These commands are translated into the register read/write operations by the Intermediate Layer, so this approach allows us to verify the implemented software layer as well.

Additionally, the controller was fully tested with comprehensive verification environment using the Universal Verification Methodology (UVM) based testbench with functional coverage metrics. This allowed complex scenarios, such as transmission errors injection or multi-device operations, to be executed. The environment is described in [8].

FPGA-based verification will be conducted to test the Bluetooth Host software stack implementation in the designed system. Development of the FPGA test platform is currently in progress.

V. Conclusion

The architecture, implementation and verification of the BLE Controller have been presented in this paper. The designed controller consists of software and hardware modules and is a part of a SoC. The design and verification issues concerning the final system functionality and its integration have been described.

For the designing process purposes it was crucial not only to design the RTL module, but also to prove that the whole system will meet the requirements. Therefore, several verification strategies have been presented. The strong points of the presented design are:

- two separate clock domains that provide easier integration with various architectures,

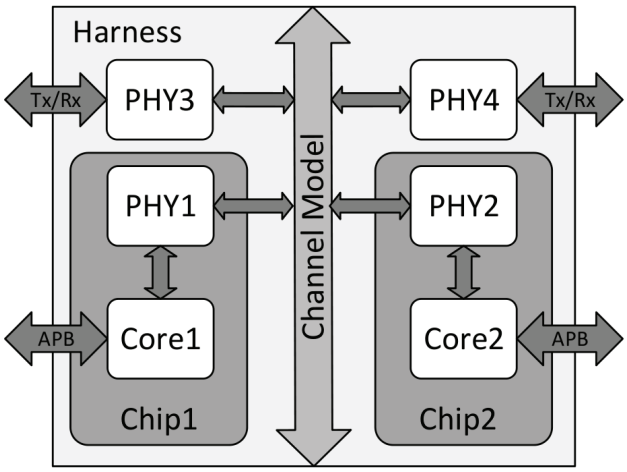

Fig. 4. Controller and PHY instances connected in the test harness.

- Intermediate Layer, which simplifies hardware part of the Controller,
- cooperation with external SPRAM for smaller chip area,
- complete verification environment using newest, advanced verification techniques,

Acknowledgment

This work was supported in part by the Polish National Centre for Research and Development under project No. DOBR/0053/R/ID1/2013/03.

References

[1] *Specification of the Bluetooth System*, Bluetooth SIG (Special Interest Group), Inc. Std., Rev. 4.1, Dec. 2013. [Online]. Available: http://www.bluetooth.org

[2] *BlueZ Official Linux Bluetooth protocol stack*, BlueZ Project, 2015. [Online]. Available: http://www.bluez.org/

[3] *BlueKitchen BTstack - Dual-mode Bluetooth stack*, BlueKitchen GmbH, 2015. [Online]. Available: http://bluekitchen-gmbh.com/

[4] I.-J. Chun, B.-G. Kim, and I.-C. Park, "A fully synthesizable Bluetooth baseband module for a System-on-a-Chip," *ETRI Journal*, vol. Volume 25, pp. 328–336, Oct. 2003.

[5] B. Yu, L. Xu, and Y. Li, "Bluetooth Low Energy (BLE) based mobile electrocardiogram monitoring system," in *IEEE International Conference on Information and Automation (ICIA)*. IEEE, Jun. 2012, pp. 763–767.

[6] *AMBA APB Protocol Specification*, ARM Std., Rev. 2.0, Apr. 2010. [Online]. Available: http://www.arm.com

[7] *Clock Domain Crossing (CDC) Design and Verification Techniques Using SystemVerilog*, Clifford Cummings Std., Rev. 1.2, 2008. [Online]. Available: http://www.sunburst-design.com/papers/

[8] M. Moskala, P. Kloczko, M. Cieplucha, and W. A. Pleskacz, "UVM-based verification of Bluetooth Low Energy controller," in *IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. IEEE, Apr. 2015, pp. 123–124.