Hardware–software design and validation framework for wireless LAN modems

C. Drosos, L. Bisdounis, D. Metafas, S. Blionas, A. Tatsaki and G. Papadopoulos

Abstract: The implementation and validation of a 5-GHz wireless LAN modem based on the HIPERLAN/2 standard is presented. In modern wireless communication systems, there is a demand for higher flexibility and more computational efficiency. Therefore the emphasis of this work is on the hardware–software structure of the developed modem and its processes, in order to offer a good balance of these requirements. In order to efficiently design and validate the behaviour of the modem, a behavioural model was developed in UML (Unified Modelling Language) as a part of the overall HIPERLAN/2 system's model. The processes of the modem were implemented in an instruction-set processor and custom hardware, combining the advantages of both software and hardware implementations. The communication between the software and hardware parts of the modem is achieved through a specialised programmable interface unit. The UML-based model of the actual HIPERLAN/2 system is used in order to validate the modem's behaviour, using scenarios from in-field usage (such as transfer of data using FTP or HTTP). Furthermore, the validation of the algorithms implemented within the modem was based on this system model, and performed through the use of a custom-validation framework. This framework produces patterns for the validation of the modem's algorithms, at three different development phases (algorithmic, HDL, FPGA-based prototyping), derived from the simulation of the system model in a consistent and automatic way. Implementation figures and co-simulation results for the developed wireless LAN modem are also given.

1 Introduction

A significant evolution in wireless office and home networks creates new opportunities for developing new products in this area providing inherent flexibility and mobility advantages. The introduction of specialised standards, such as the ETSI BRAN HIPERLAN/2 [1] and the IEEE 802.11a [2], has opened the road to new products in this area. It has been estimated that more than 61 million wireless products will be shipped by 2006 [3]. Both aforementioned standards operate in the 5 GHz band, and both utilise orthogonal frequency division multiplexing (OFDM) for multicarrier transmission [4].

The HIPERLAN/2 standard features many physical layer data rates, with a maximum of up to 54 Mbps, using a combination of modulation schemes and coding factors [5]. Furthermore, it supports advanced quality of service (QoS) features for streaming audio and video services [6]. The QoS features are provided by the medium access control (MAC) layer of the protocol, through a central resource control scheme, which controls the link capability among access point and mobile terminals according to interference situations and distance.

The scope of this paper is to present the design and implementation of a modem for the HIPERLAN/2 standard. The complicated nature of a HIPERLAN/2-based system (real-time performance, area, power, flexibility and QoS requirements) suggests the usage of dedicated processors, for the upper protocol's layers, in close collaboration with the modem's functions. In order to design the modem itself along with its complicated and flexible interface with the upper layers, a novel design flow was followed. The flow begins with the modelling of the behaviour of the modem and its interface as parts of the executable specification model of the overall wireless system, using unified modelling language (UML) [7] for this purpose. In this way, validation of the interface and basic modem's functionality is performed at a high level of abstraction, along with the rest of the system, using system-level usage scenarios. The use of high-level modelling in the early phase of the design helps to eliminate design flaws and allows the thorough examination and evaluation of the interface of the modem.

The development flow that was followed for the development of the HIPERLAN/2 system and its modem is presented in Fig. 1. After obtaining the specifications, the system is modelled using UML. The model is executed to validate the conformance of the modelled system to its specifications, and the hardware and software design phases follow. After the design phases, the system artefacts are validated once more, prior to the final system-integration and validation phase. The validation of the hardware parts of the system is integrated with the validation at the system level through the specially developed validation framework, which is presented in Section 5.

IEE Proceedings online no. 20040496

doi: 10.1049/ip-cdt:20040496

Paper first received 17th December 2003 and in revised form 1st March 2004

C. Drosos, L. Bisdounis, D. Metafas, S. Blionas and A. Tatsaki are with INTRACOM S.A., 19.5 Km Markopoulo Ave., P.O. Box 68, GR-19002 Peania, Athens, Greece

G. Papadopoulos is with Applied Electronics Laboratory, Department of Electrical and Computer Engineering, University of Patras, 26500 Patras, Greece

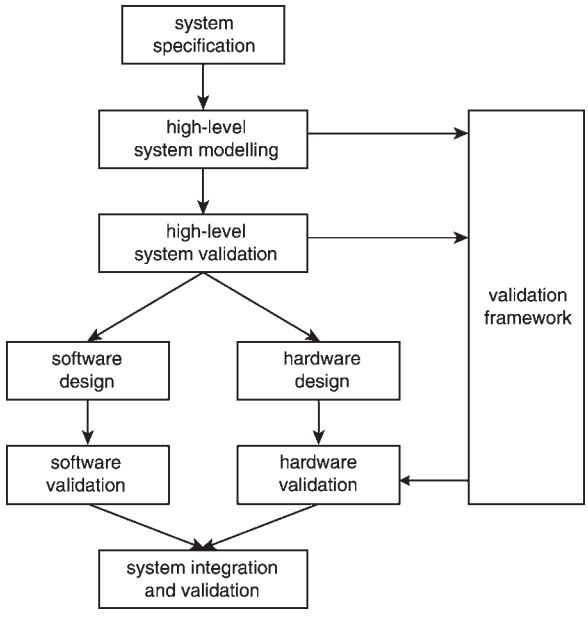

Fig. 1 *Hardware–software design and validation flow*

The implementation of the modem is divided into software running in a dedicated instruction-set processor and custom hardware modules. This implementation strategy combines the advantages of hardware implementation (high speed, low power dissipation) with the efficiency that derives from the software implementation of the modem's critical parts, such as the central resource manager (scheduler) of the MAC layer. The seamless communication between the modem's processes implemented in the software and the hardware can be guaranteed only by a specialised interface, which was designed and validated thoroughly through the high-level model of the system.

The high-level modelling approach is also used for the validation of the modem's algorithms. This approach is based on a validation framework, which takes as input a high-level description of HIPERLAN/2 frames and automatically produces test patterns for the validation of the design of the modem in three different phases. The three phases used for the validation are the algorithmic theoretical development phase, the design phase (VHDL) and the implementation phase (FPGA). The validation approach, used during the development of the modem, integrates the validation of the modem's design (interface, algorithms) with the protocol's validation.

Concerning previously proposed mode implementations at the 5-GHz band; in [8] the baseband processing of both 5-GHz standards (HIPERLAN/2, IEEE 802.11a) is implemented by using a combination of a custom digital signal processor and a custom streaming coprocessor, whereas in [9] ASIC implementations of the OFDM modem are proposed. Both cases do not present the interface with the upper layers of the protocol stack. A major advantage of this paper is that it introduces a detailed interface between the modem and the protocol, which has been developed and validated through exhaustive simulation at different levels of abstraction. The emphasis of this paper is mainly on the hardware and software codesign aspects of the modem, the communication between its hardware and software parts as well as on the validation methodology. The detailed description of the modem's and the overall WLAN system's hardware is beyond the scope of this paper, and more details can be found in [10] and [11].

2 High-level modelling of the modem

The major problem in the design of a complicated system, such as the wireless LAN modem, is the verification of the design and the early detection of design faults [12]. For this reason, a UML-based flow for the system design of the modem was followed in order to produce an executable system specification (virtual prototype) for early verification of its control functions. This virtual prototype is based on a UML model, which uses the UML-RT profile [13], based on the real time object-oriented modelling (ROOM) methodology [14]. This UML profile is capable of representing the architecture of the system at a high level of abstraction [15].

The design flow starts with obtaining the system's specification. During this phase, the specifications of the system are formally obtained using the modeling techniques offered by UML, such as the use-case and sequence diagrams. An example of a use-case diagram that models some of the services of the physical layer of the protocol stack is presented in Fig. 2. The internal structure of the modem is then shaped, and is modelled as abstract objects and classes. In this way, a UML model is created, which contains the main functional blocks of the control processes of the modem, not committed to a specific implementation strategy. The behaviour of each process is then captured inside the objects of the model, in a pure behavioural way, using state charts [16] provided by UML to create abstract finite state machines.

In addition to the modem's functionality, the UML model also contains the protocol stack of the HIPERLAN/2 [1] system, in order to complete the modelling of the overall system. The produced virtual prototype of the system, in terms of executable system specifications, can be used for the early verification of the modem and its programmable interface. This phase is as important as the overall implementation of the modem itself [12]. The verification of the modem is performed against the usage scenarios of the system that were identified when the specification was obtained. As a result, the actual HIPERLAN/2 protocol stack implementation, modelled in UML, is used to verify the modem's behaviour using scenarios from in-field usage, such as the association and connection setup procedures and the transfer of data using FTP or HTTP.

The structure of the UML model that was used for the co-simulation and verification of the modem is presented in Fig. 3. The developed modem consists of three main building blocks: the access point (AP) model, the mobile terminal (MT) model, the ethernet models and the tester model, which is the core of the testbench and a model of the air interface. The tester model is capable of introducing

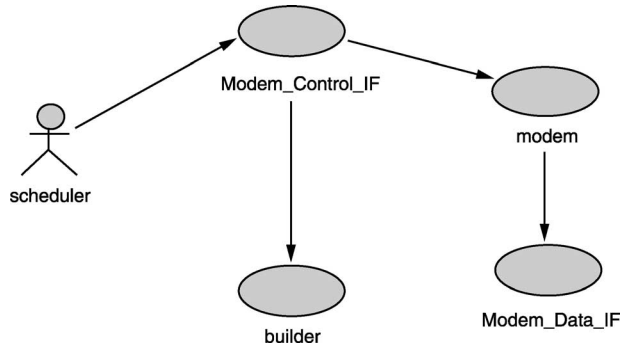

Fig. 2 *Use-case diagram for the physical layer of the access point*

random errors inside the physical layer bursts, simulating, in this way, the real-world behaviour. The AP and MT models consist of the UML-based behavioural model of the modem, and the HIPERLAN/2 protocol stack, suitably modified for the AP or MT specific needs, respectively. The tester allows the communication between the two network nodes (AP and MT) in the form of a physical-layer burst. Apart from these blocks, the testbench also contains models for sources and sinks of ethernet packets. Each network node, AP or MT, is connected to one source and one sink of ethernet packets, to simulate the traffic generated by the core network.

The detailed structure of the AP model is presented in Fig. 4, as a UML structure diagram. This diagram presents the objects that comprise the high-level AP model. Each object is associated with a state machine, describing the behaviour of the corresponding object's functionality. The structure presented in this Figure is the same for the MT model, apart from the scheduler object, which does not exist inside the MT model.

Within the structure of the AP model there is a group of three objects that model the interface of the modem with the rest of the system, as well as the behaviour of the modem's control parts. These objects are the 'modem data IF', the 'modem control IF', and the 'modem bursts creation'. The first object models the behaviour of the data manipulation parts of the modem's interface (memories and registers), the second object models the behaviour of the control parts of the interface (commands accepted by the modem, interrupts), whereas the third object models the behaviour of the modem (creation of broadcast, downlink and uplink data bursts).

The other important group of objects models the interface of the rest system with the modem. The HIPERLAN/2 standard does not specify this interface, and so the resulting scheme presented here is the outcome of the co-simulation of the system with the high-level UML model. The objects responsible for the interface of the protocol with the modem are the 'AP scheduler', the 'AP frame builder' and the 'AP frame decoder'. The scheduler is the part of the interface that shares the resources of the physical layer among the various DLC (data link control) connections and their QoS requirements, producing the format of each HIPERLAN/2 frame. The format of the frame is then translated into modem commands, stored inside the 'modem control IF' object. The frame builder is responsible for the creation of the downlink bursts of the frame for the AP. Its input is the format of the downlink burst produced by the scheduler, and its operation is to collect transmission packets from the DLC queues and format them accordingly, transferring the resulting parts of the frame to the 'modem data IF' object. The operation of the decoder object is the equivalent for the uplink parts of the frame. An important feature of the proposed implementation of the interface parts of the modem with the rest system (DLC layer of the protocol) is that the overall scheme operates without intermediate packet transfers. The operation of the interface objects is based on the exchange of pointers, minimising this way the utilisation of the system bus. The only packet transfers of the system are between the DLC queues and the modem, and the DMA (direct memory access) engine of the system performs them.

The objects outside the grey area in Fig. 4 concern the DLC layer (data transport functions [17] and radio link control sublayer [18]) as well as the ethernet specific convergence layer (ECL) [19] of the HIPERLAN/2 standard, which are beyond the scope of this work.

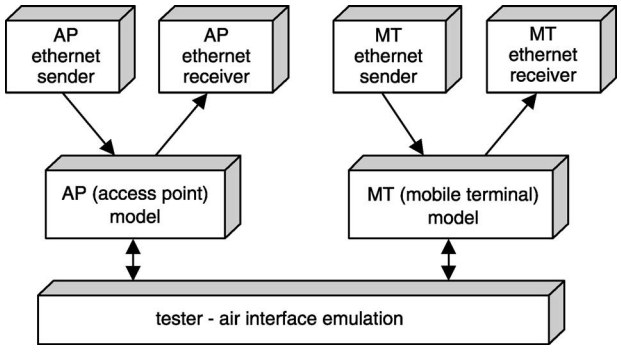

Fig. 3 *High-level model of the modem*

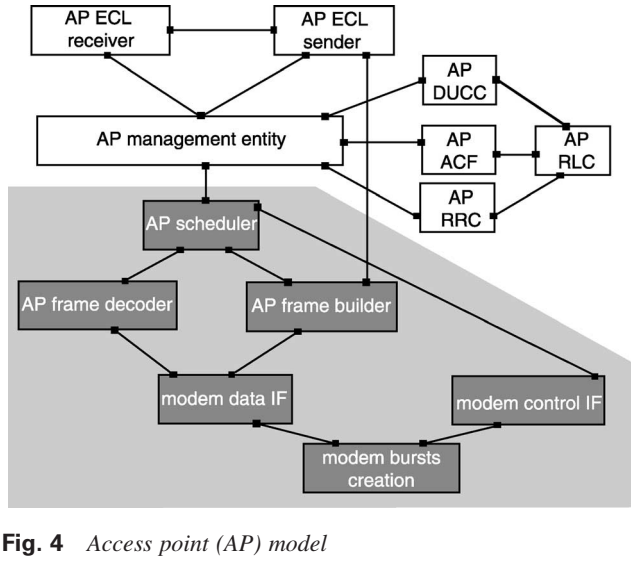

Fig. 4 *Access point (AP) model*

3 Hardware–software mapping and communication

An important task during the design of the modem is the mapping of its processes to the used instruction-set processors, i.e. software implemented processes, and to custom hardware, i.e. hardware implemented processes, as well as the determination of the interface of the processes implemented in the software with those implemented in the hardware. The mapping scheme that is used in this development is heuristic (based on our previous experience from the development of similar systems), as in the majority of today's systems [20]. The architecture of the platform (ARM integrator [21]), in which the modem's prototype is implemented, was selected in order to realise the produced mapping scheme of the overall system. The prototype platform mainly consists of two ARM processor cores [22] for the realisation of the processes that were mapped for implementation in the software, and FPGA devices for the implementation of the hardware parts of the modem. The processor cores are responsible for the execution of the upper layers of the HIPERLAN/2 protocol stack and the software parts of the modem. The physical layer of the protocol (the modem's data path and control units) was implemented onto the FPGA devices of the prototype platform. Details of the architecture of the overall system implementing the HIPERLAN/2 wireless LAN standard can be found in [11].

Table 1 summarises the processes of the modem that are implemented in the software along with those implemented in the custom hardware. For the implementation of the interface between the modem and the HIPERLAN/2 DLC layer, a fully programmable hardware unit has been designed, capable of executing a set of suitable instructions.

The selection of the instruction set of the modem is based on the results of alternative instruction sets using the high-level virtual prototype of the system. Using this prototype, the partitioning scheme was also validated and verified in order to prove that it meets the requirements of the system's functionality. Details of the architecture and the implementation of the programmable modem interface unit are given in the next Section, where the modem's implementation is presented.

The reason for implementing some objects of the modem in the software is the increased flexibility that the design maintains with this solution. Concerning the scheduler of the system, a software implementation allows the modification of its algorithms to enhance its operation by introducing newer and more efficient ways to share the wireless bandwidth and to meet QoS requests. The scheduler executes an algorithm that categorises the connections to QoS classes, and shares the capacity of the frame to the classes, using fixed priorities. Furthermore, in extreme situations, it can dynamically modify the priorities to allow maximum utilisation of the frame and to avoid starvation of the connections of the lower class [23]. The close collaboration of the scheduler with the frame builder and decoder processes was also the reason for the implementation of both objects into the software. Furthermore, in order to maintain the flexibility that is offered by the software implementation, a separate processor was selected for the mapping of the software parts of the modem interface, in order to keep its independence from the other layers of the protocol stack that are running on a different processor [11].

The remaining processes of the modem and its interface, presented in Table 1, were implemented in the custom hardware, since this solution offers the implementation speed required for such a complex functionality. Apart from the modem's interface block, the remaining blocks of the right column in Table 1 belong to receive and transmit paths of the baseband modem. The structure of both baseband modem paths is presented in the next Section, where the architecture and the modem's implementation are presented.

Table 1: Mapping of the modem's processes and interface to the instruction-set processor and custom hardware

Processes implemented in software	Processes implemented in hardware
Scheduling	data scrambling and descrambling
Frame building	forward error correction (FEC) encoding and decoding
Frame decoding	data interleaving and deinterleaving
	constellation encoding and decoding
	direct and inverse Fourier transform (FFT and IFFT)
	Symbol synchronisation and frequency offset estimation and correction
	burst formation
	pilot insertion and equalisation
	cyclic prefix insertion and extraction
	frequency equalisation
	channel estimation
	modem interface processes (programmable unit)

4 Modem implementation

The implementation of the modem has two different aspects: the first concerns the software parts, whereas the second deals with the custom hardware implementing some parts of the modem, see Table 1. The implementation of the modem's software parts was based on the high-level model of the system (Section 2), which was further refined to include the implementation details of the three objects that were implemented in the software (scheduler, frame decoder, frame builder). The functionality of these modem's objects was designed using state-charts, e.g. the one presented in Fig. 5, for the frame builder block of the AP. The development of the software parts of the modem was performed using the Rose-RT tool [24], and automatic executable source code generation from the UML model of the corresponding objects.

Table 2 presents average execution times for the important functionalities of the modem's software objects during a frame with typical traffic. The scenario uses the QPSK modulation scheme [5] with a code rate of 3/4 (maximum throughput = 18 Mbps) and 75% utilisation of the frame duration (total frame duration = 2 ms). The operating frequency for the ARM processor running the code was 50 MHz. Furthermore, using the 64-QAM modulation scheme with a code rate of 3/4 [5], the maximum available user throughput of the system reaches 42 Mbps.

Both receive and transmit paths of the modem (Fig. 6) were implemented in the hardware, using a pure VHDL flow. The transmit path of the modem accepts binary input data, which are then scrambled, encoded by the convolution encoder. Interleaving then takes place, followed by modulation, using QPSK or QAM schemes. The forward error control block then processes the resulting data, which are fed to the IFFT block that transforms them to the time

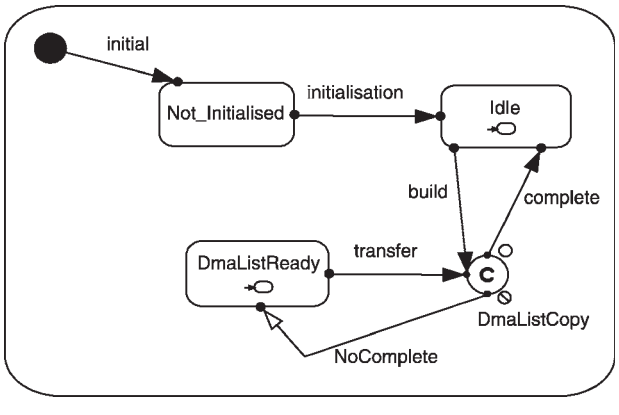

Fig. 5 *State-chart for the builder object of the AP*

Table 2: Implementation figures for the modem's software parts

AP functionality	Execution time (μs)	Source code size (lines)	Binary code size (KB)
Scheduler	240	5600	59.6
Frame builder	320	11800	77.8
Frame decoder	370	9800	63.7

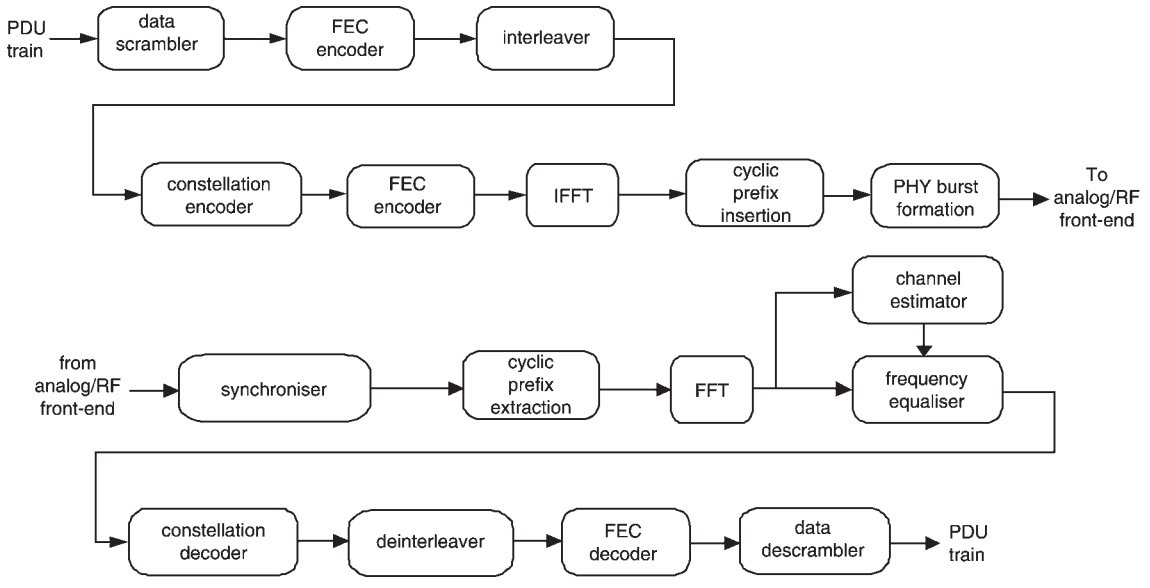

Fig. 6 *Modem's transmit and receive paths*

domain. A cyclic prefix is then added, to compensate for multi-path fading, and finally the burst of the physical layer is created.

In the receive path of the modem, after the synchronisation (symbol synchronisation and frequency offset estimation and correction), the cyclic prefix extractor removes the extra samples, and then the FFT block transforms the signal back to the frequency domain. The channel estimator unit then estimates the phase and amplitude attenuation, and the frequency equaliser performs the equalisation of the received signal. In the data domain of the receive path, the decoder decodes the data, which are then deinterleaved and processed by the forward error control (FEC) decoder. Finally, the descrambling of the decoded data bits is performed.

Another important part of the modem is the implementation of its communication with the upper layers of the HIPERLAN/2 protocol stack [1]. The modem's interface is, in fact, the interface between the ARM microprocessor and the baseband modem. It features a fully programmable engine by the ARM, which generates all the control information for the baseband receiver and transmitter. Furthermore, it also handles the received data, which are forwarded to the processor and the transmitted data and sent to the modem by the ARM processor.

The interface of the modem's hardware with the software parts of the modem includes a number of memory buffers, a number of interrupt signals and a command set. The command set of the modem, along with a short description of each command is summarised in Table 3. Apart from the commands, the interface unit includes a number of storage buffers (e.g. command memory, transmit and receive data memories) and a series of interrupts (e.g. synchronisation, end of receive, end of transmit), as acknowledgements to the protocol's requests.

The internal architecture of the modem interface unit illustrating its basic blocks is given in Fig. 7. The 'command translator' block generates the control signals for both transmit and receive data paths of the modem. The 'puncturing' block generates the puncturing control signal indicating when puncturing insertion should happen. The 'scrambler initialisation' block generates the control signals to initialise the data scrambler and descrambler of the modem. Scrambler initialisation happens in the beginning of each burst of data or after the change of the link mode within a burst. The 'memory address generator' block generates the address and the necessary control signals to access both receive and transmit memories. The 'interrupt generator' block generates the required interrupts for the

Table 3: List of the modem's interface commands

Command	Description
Tx	transmits a downlink burst (arguments = packet type, number of packets, physical mode, time slot)
Tx_S	transmits an uplink burst with short preamble (arguments = packet type, number of packets, physical mode, time slot)
Tx_L	transmits an uplink burst with long preamble (arguments = packet type, number of packets, physical mode, time slot)
Rx	receives a downlink burst from the mobile terminal (arguments = packet type, number of packets, physical mode, time slot)
Rx_S	receives an uplink burst with a short preamble (arguments = packet type, number of packets, physical mode, time slot)
Rx_L	receives an uplink burst with a long preamble (arguments = packet type, number of packets, physical mode, time slot)
IQ_EN	turns on/off the RF I/F of the board (arguments = time slot)
RESET	resets the modem's pointers to its memories to their initial position, sets the number of frames for synchronisation search, sets the slot number after synchronisation
END	flushes out the transmit of the receive pipeline (arguments = time slot)
NOP	the modem does not perform any operation (arguments = time slot)
BCH_SRCH	searches for the synchronisation preamble
CFG	configures the modem with the contents of the configure memory (arguments = time slot)

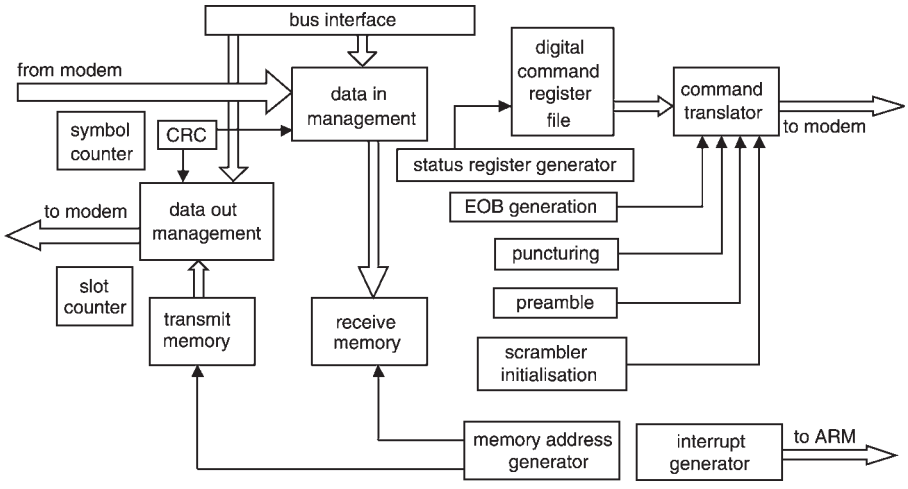

Fig. 7 *Internal architecture of the modem's interface unit*

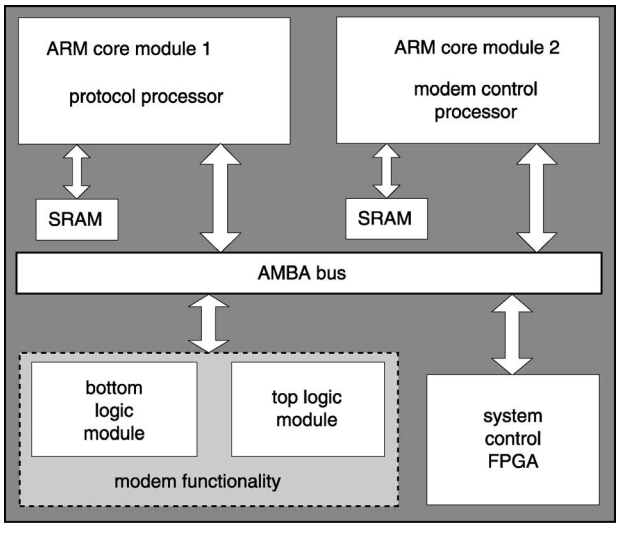

Fig. 8 *Block diagram of the used ARM integrator platform*

ARM processor. The 'bus interface' block is a slave bus interface for the AMBA system bus [25, 11], and communicates with the modem's memories. The 'slot counter' controls the execution time of each control command. The modem interface unit parses the control commands and the configuration information, and generates the necessary control information for the baseband receiver and transmitter.

The hardware parts of the modem and its interface were implemented onto two logic modules [26] of the ARM integrator platform [21], each one featuring a XILINX Virtex E 2000 FPGA [27] with 500 K usable gates and 640 KB additional RAM. The average utilisation of the FPGAs is 87%. An abstract diagram of the platform is given in Fig. 8.

More details on the realisation of the modem on the logic modules of the ARM integrator platform can be found in [10] and [11], since the emphasis of this work is on the investigation of the combined hardware–software design aspects of the modem as well as on its validation methodology.

5 Validation framework and the modem's validation

The validation of the design is performed in three different phases. The first one concerns the validation at the algorithmic level, where both receive and transmit algorithms of the modem are developed and verified at the numerical level. The second phase concerns the VHDL development environment and its simulator, whereas the third validation phase concerns the reconfigurable hardware and its accompanying platform.

Before the validation of the modem's algorithms, for the purposes of system validation, an executable specification model of the system has been developed, using UML [7]. This model helps the validation of the specifications of the system early in the design process. An important artifact of the UML modelling is a set of system-usage scenarios, captured formally using the UML's modelling techniques. These usage scenarios, accompanied with sequence diagrams formally presenting each one, are a helpful resource for the validation of the system's design at various levels of abstraction. This model is also used to validate the design of the system's software, using as input the formally obtained system specifications in the form of sequence diagrams.

One approach to the validation of the hardware design of the system, and more specifically the modem, is to use the formal validation techniques offered by UML modelling for this purpose, and the developed UML model of the system. The idea is to use the same sequence diagrams and the same UML system's model for the validation of the hardware blocks of the modem, suitably modified to meet the nature and specific requirements of hardware validation. The way to achieve this is to create a methodology and a validation framework that takes as its input usage scenarios from the system level and produces validation patterns for all the three modem's validation phases. In order for the validation to be accurate, the transformation of the high-level usage scenarios to low-level test patterns for the validation of the modem has to be automatic and formal.

In order to automate and facilitate the method for generating the test input for all three phases of the modem's validation from the high-level system scenarios, a custom validation framework was developed. This framework provides an automatic connection of the whole system's validation with the validation of the modem at the various phases of its development process. The operation of the developed framework is presented graphically in Fig. 9. Its major advantage is that from a given high-level frame description it can produce test patterns for the three validation phases that are consistent, and perform the same validation scenario at the corresponding validation environment.

The input of the validation framework is descriptions of the map of a HIPERLAN/2 frame at a high-level format.

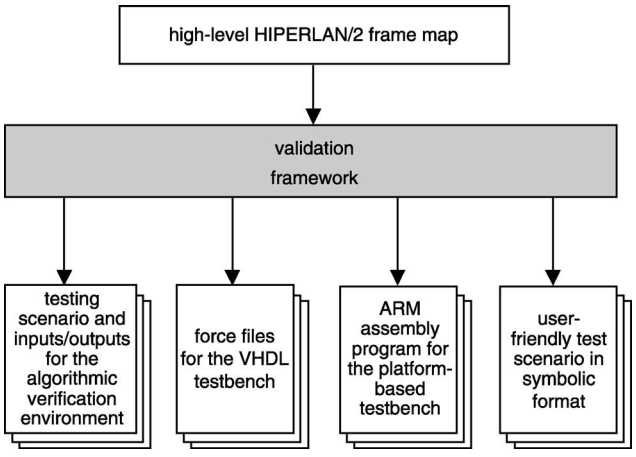

Fig. 9 *Input-output artifacts of the validation framework*

These descriptions are obtained during the execution of the UML model of the system, under the stimulus of the usage scenarios, in the form of sequence diagrams. During the execution of these testing scenarios by the UML system model the outputs of the scheduler class of the system are logged. These logs contain the maps for a selected number of MAC frames, as these maps have been generated by the central scheduler of the system and its policies for allocating transmission resources. The validation framework is able to process these types of high-level frame information to produce test patterns for all three phases of the modem's validation.

The first phase of the validation of the modem concerns the validation of the receive and transmit algorithms in a MATLAB environment [28]. An ad-hoc testbench has been created inside this environment to validate the developed algorithms. This testbench can be driven by a simulation scenario file and data files containing data to be processed by the algorithms. All these types of files are generated automatically by the validation framework, to efficiently validate the algorithms of the modem with the same operating scenarios as the ones used for the validation of the system's UML model.

The second validation phase supports the VHDL-based design of the modem. Inside the context of this phase, a VHDL-based testbench has been developed to help the verification task. The role of the validation framework during this phase is to produce input files and to support the operation of this VHDL-based testbench. More specifically, the framework produces force files that fill the memory blocks of the modem (configuration, transmit and control command memories) with valid contents, according to the high-level frame description. Furthermore, inputs and outputs of each algorithmic block are produced during the validation of the algorithms at the numerical level (MATLAB environment), which can also be used for the verification of the VHDL-based design, in direct comparison with the algorithmic results.

The last phase of the modem's validation concerns the FPGA-based prototype platform. The platform contains the reconfigurable blocks that realise the modem's circuitry and the system's processors. The validation phase at this platform is performed through the hardware–software interface of the modem, and is driven by the processor that is responsible for the modem control (Fig. 8). The role of the validation framework during this phase is to automatically produce the processor source code that will drive the validation, by using the high-level frame description. The code must fill all the modem's memories with valid data specific to the validation scenario and control the execution of this scenario.

In order to present the detailed operation of the validation framework and the functions that it performs in the context of the verification of the system's algorithms, results from its application to the verification process of the wireless LAN modem are given below. The modem verification case presented here is based on an in-filled usage scenario, suitably modified to meet the specific validation environments used for the development of the algorithms. The scenario used for the testing of the modem design is a part of the protocol's radio link control (RLC) association scenario, between the AP and the MT. A number of protocol frames are captured during this high-level usage scenario and are translated into suitable test vectors for the verification of the modem. One of these frames is presented in Fig. 10 showing the bursts comprising a protocol frame, along with the starting time slot, duration and number of data units for each burst.

The presented frame includes the broadcast burst (access point identity, capabilities and map of the current frame), the downlink burst (which contains a train of 5 short (SCH) and 12 long (LCH) packets), and the uplink burst (1 long and 10 short packets). Furthermore, the frame includes a random access burst, during which unassociated mobile terminals can perform their initial contact with the access point.

The structure of a protocol frame, captured from high-level system verification, can be translated using the developed framework into a number of test vectors for every development phase of the modem. Apart from the creation of the test vectors, the validation framework produces a representation of the testing scenario in a symbolic format, using the modem instructions. The automatically produced modem program, in the form of symbolic commands, for the aforementioned protocol frame is presented in Fig. 11.

Apart from the modem command program in symbolic format, the validation framework produces test vectors for the MATLAB testbench, the VHDL-based testbench and the FPGA platform. For the presented testing scenario, the full contents of all the memory buffers of the modem for both AP and MT nodes are produced, along with validation control loops. The operation of the testbench is to fill the buffers of the modem's interface with suitable input values and to drive the inputs of the modem with valid data streams. In the case of the FPGA-based prototype platform, the validation framework produces a testbench for the modem's dedicated processor, in the form of an ARM

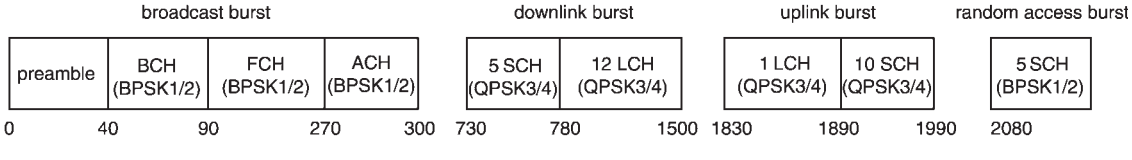

Fig. 10 *HIPERLAN/2 frame for modem validation*

AP symbolic program

```
CFG(4900, 1)
NOP(4901, 1)
RESET(0, 4949, 1)
TX(BCH_P1, 1, BPSK12, 4950, 1)
TX(FCH_P1, 1, BPSK12, 0, 1)
TX(FCH_P1, 1, BPSK12, 90, 1)
TX(ACH_P1, 1, BPSK12, 180, 1)
END(0, 1, 0, 210, 1)
TX(SCH_P1, 1, QPSK34, 560, 1)
TX(SCH, 4, QPSK34, 570, 1)
TX(LCH, 12, QPSK34, 610, 1)
END(0, 1, 0, 1330, 1)
IQ_EN(1, 0, 1530, 1)
RX_S(LCH, 1, QPSK34, 1700, 1)
RX_S(LCH, 1, QPSK34, 1735, 1)
RX_S(SCH_P1, 1, QPSK34, 1795, 1)
RX_S(SCH, 9, QPSK34, 1805, 1)
END(1, 0, 0, 1896, 1)
RX_S(SCH, 1, BPSK34, 1950, 1)
RX_S(SCH, 1, BPSK34, 1985, 1)
END(1, 0, 0, 2035, 1)
IQ_EN(0, 0, 2036, 1)
```

MT symbolic program

```
CFG(4780, 1)
NOP(4781, 1)  {not used for VHDL testbenches}
RESET(0, 39, 1)
IQ_EN(1, 0, 4790, 1)
NOP( 4920, 1)  {not used for VHDL testbenches}
BCH_SEARCH( 1)
RX_L(BCH_P1, 1, BPSK12, 40, 1)
RX_L(FCH_P1, 1, BPSK12, 90, 1)
RX_L(FCH_P1, 1, BPSK12, 180, 1)
RX_L(ACH_P1, 1, BPSK12, 270, 1)
END(1, 0, 0, 300, 1)
RX_DL(SCH, 1, QPSK34, 710, 1)
RX_DL(SCH_P1, 1, QPSK34, 730, 1)
RX_DL(SCH, 4, QPSK34, 740, 1)
RX_DL(LCH, 12, QPSK34, 780, 1)
END(1, 0, 0, 1500, 1)
IQ_EN(0, 0, 1501, 1)
TX(LCH, 1, QPSK34, 1750, 1)
TX(SCH_P1, 1, QPSK34, 1810, 1)
TX(SCH, 9, QPSK34, 1820, 1)
END(0, 1, 0, 1910, 1)
TX_S(SCH, 1, BPSK34, 2000, 1)
END(0, 1, 0, 2050, 1)
```

Fig. 11 *Modem program in symbolic format (produced by the validation framework)*

assembly code. The role of this code is to program the modem and to inspect the results that it produces.

6 Validation results

In order to exploit the advantage offered by a pure VHDL design and simulation environment, the validation framework has been developed and utilised. This framework converts the high-level frame description into a set of force files in a Modelsim format. Since the Modelsim tool [29] cannot handle the format of the frame and therefore cannot fill the memories of the modem interface unit (configuration, transmit and control command memories) with their contents, in order for the baseband modem to be programmed, the force files must be produced.

In the timing diagrams of Fig. 12, the signal 'AP_Tx_IQ_data_out' and 'MT_Tx_IQ_data_out' are the transmitter's output for the access point and the mobile terminal, respectively. The signals 'AP_Rx_data_out' and 'MT_Rx_data_out' are the receiver's output (the data that are inserted in the receive memory of the modem interface unit) for the access point and the mobile terminal, respectively. Also, the basic control signals for both the access point and mobile terminal are shown ('data_valid', enable and control commands for the receiver and start of transmission for the transmitter). Note, that analog interpretation of the data signals is used for better understudying of the results.

Figure 12 illustrates the simulated signals for a complete frame (broadcast, downlink and uplink bursts). These signals were generated after the execution of the simulation scenario that was produced automatically by the validation framework for the input frame given in Fig. 10. The different bursts of the simulated frame are clearly presented in Fig. 12. For each burst of the frame, the packets that it contains are also displayed in the same Figure. For the transmission of the user packets (LCHs), this example uses the QPSK 3/4 physical mode. Similar test cases were also performed in order to cover all the physical layer modes supported by the HIPERLAN/2 standard [30].

7 Conclusions

This paper presents the implementation and the validation of the software and hardware parts of the baseband modem for a wireless LAN application. For the design of the modem, a flexible implementation scheme was followed, where some parts of the modem were implemented in hardware, for efficiency reasons, and some parts in software, for flexibility reasons. The specialised communication between the hardware and the software parts of the modem was designed and validated by using the high-level model of the system.

The validation of the modem and its interface was based on a high-level system model, developed using UML. This UML-based model of the actual HIPERLAN/2 system was used, in order to validate the modem's behaviour using scenarios from in-field usage (such as the association and connection setup procedures and the transfer of data using FTP or HTTP). The same approach was also used for the validation of the algorithms implemented within the modem. For this purpose, a validation framework was developed. The framework uses as input a high-level frame format, as it has been produced by the simulation of the UML system model, and produces specific test patterns for the validation of the modem's algorithms, at three different phases of the development (algorithmic, HDL, FPGA-based prototyping).

The main benefit of the validation framework presented in this paper is that it integrates the validation of the modem's development with the validation of the rest of the system, which is performed at a high level of abstraction using UML modelling. This way, initially the behaviour of the modem can be validated inside the UML model of the system, using real-usage case scenarios for this purpose, and then, the implementation of the modem and its algorithms can be performed using the validation framework that integrates the validation at this level with the validation at the system level. The use of this validation framework helps in the easy and efficient identification and correction of the modem's design errors. Furthermore, the proposed validation framework is based on the modelling of the system

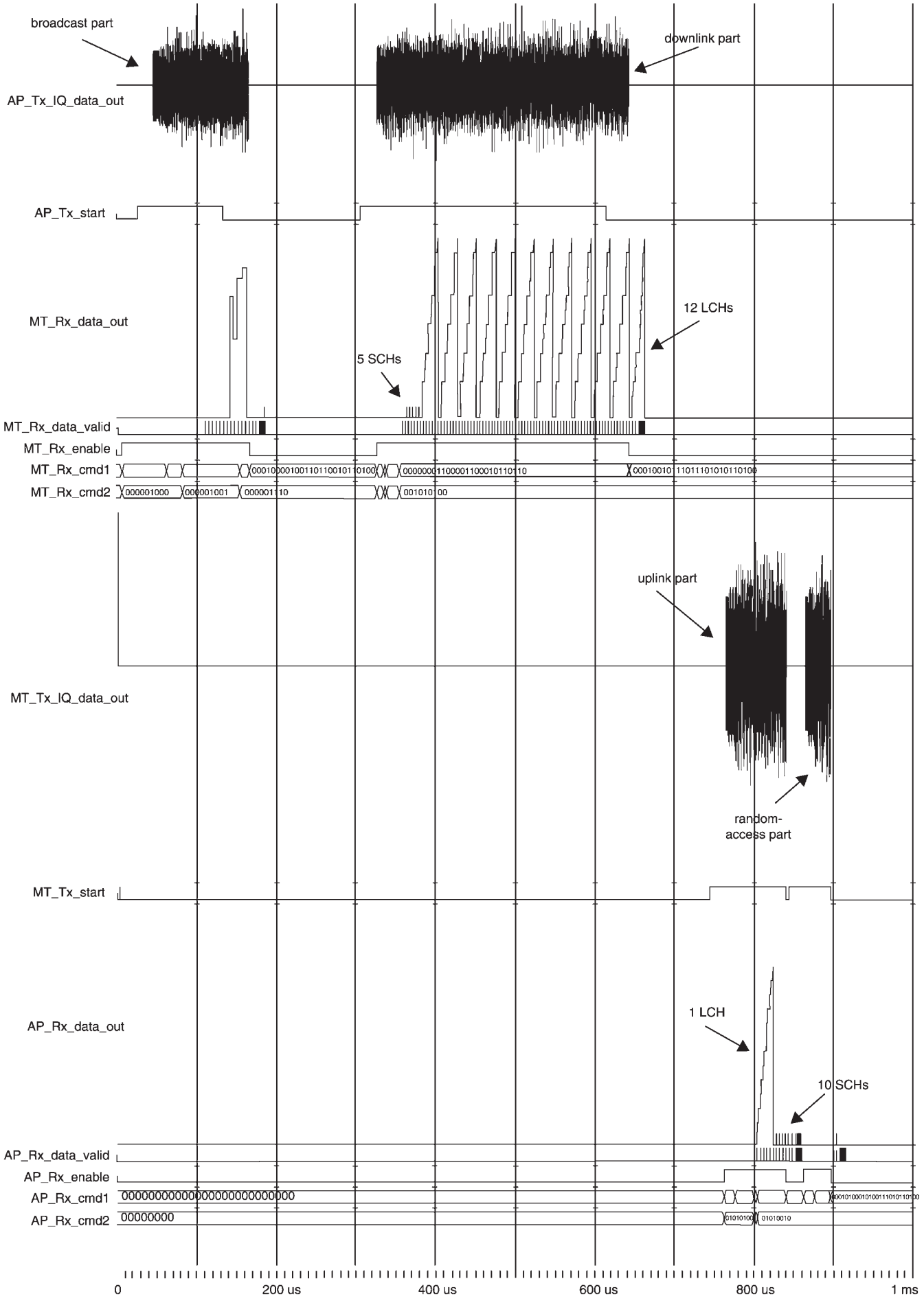

Fig. 12 *Timing simulation of the modem using the VHDL testbench*

with UML, which is becoming a trend in today's embedded systems [15].

The proposed validation framework, along with UML modelling at a high level, helped the design of the HIPERLAN/2 modem by speeding up the system-integration and validation phases. The validation support that provided the framework and high-level modelling was really important for the efficient and fast development of the modem, and therefore of the overall system. Other proposed methodologies for telecommunication systems (such as the one presented in [8]) do not offer support to the validation of the modem's design at different levels of design abstraction (system model, mathematical analysis, HDL design, FPGA prototype), even if they propose UML for the modelling of the system, as in the design methodology presented in [31]. The use of UML for the modelling and validation of the modem as has been proposed in this paper significantly boosts its development.

The prototype of the presented modem was implemented successfully on an FPGA-based platform (ARM integrator). The characteristics of the implemented software and hardware parts of the modem are presented throughout the paper. Furthermore, results from the co-simulation of the modem through the testing scenarios produced by the validation framework are also presented, along with sample outputs of the framework.

8 Acknowledgments

This work was partially supported by the project IST-2000-30093 EASY (energy-aware system-on-chip design of the HIPERLAN/2 standard), funded by the EU. The authors also thank the reviewers for their valuable suggestions and comments in order to improve the paper's structure and research results.

9 References

1 European Telecommunication Standards Institute (ETSI), broadband radio access network (BRAN), 'HIPERLAN Type 2: system overview', TR 101 683, February 2000
2 The Institute of Electrical and Electronics Engineers (IEEE), 'Supplement to IEEE standard for information technology – telecommunications and information exchange between systems – local and metropolitan area networks – specific requirements – part 11: wireless LAN medium access control (MAC) and physical layer in the 5 GHz band', IEEE Std. 802.11a, 1999
3 iSuppli Corp., 'The future of wireless connectivity: Bluetooth and WLAN'. Market analysis report, Q3 2002
4 Van Nee, R., and Prasad, R.: 'OFDM for mobile multimedia communications' (Artech House, Boston, MA, 1999)
5 European Telecommunications Standards Institute (ETSI), broadband radio access network (BRAN), 'HIPERLAN Type 2: physical (PHY) layer', TS 101 475, April 2000
6 Doufexi, A., Armour, S., Butler, M., Nix, A., Bull, D., and McGeehan, T.: 'A comparison of the HIPERLAN/2 and IEEE 802.11a wireless LAN standards', *IEEE Commun. Mag.*, 2002, **40**, (5), pp. 172–180
7 Booch, G., Rumbaugh, J., and Jacobson, I.: 'The unified modeling language user guide' (Addison-Wesley Professional, 1998)
8 Kneip, J., Weiss, M., Drescher, W., Aue, V., Strobel, J., Oberthur, T., Bolc, M., and Fettweis, G.: 'Single chip programmable baseband ASSP for 5 GHz wireless LAN applications', *IEICE Trans. Electron.*, 2002, **85**, (2), pp. 359–367
9 Eberle, W., Derudder, V., Vanwijnsberghe, G., Vergara, M., Deneire, L., Van der Perre, L., Engels, M.G.E., Bolsens, I., and De Man, H.: '80-Mb/s QPSK and 72-Mb/s 64-QAM flexible and scalable digital OFDM transceiver ASICs for wireless local area networks in the 5-GHz band', *IEEE. J. Solid-State Circuits*, 2001, **36**, (11), pp. 1829–1838
10 Blionas, S., Masselos, K., Dre, C., Drosos, C., Ieromnimon, F., Metafas, D., Pagonis, T., Pneymatikakis, A., Tatsaki, A., Trimis, T., and Vontzalidis, A.: 'Prototyping of a 5 GHz WLAN reconfigurable system-on-chip', *IEICE Trans. Inf. Syst.*, 2003, **E86-D**, (5), pp. 901–909
11 Bisdounis, L., Dre, C., Blionas, S., Metafas, D., Tatsaki, A., Ieromnimon, F., Macii, E., Rouzet, Ph., Zafalon, R., and Benini, L.: 'Low-power system-on-chip architecture for wireless LANs', *IEE Proc., Comput. Digit. Tech.*, 2004, **151**, (1), pp. 2–15
12 Sangiovanni-Vincentelli, A., McGeer, P., and Saldanha, A.: 'Verification of electronic systems'. Proc. Conf. on Design Automation, Las Vegas, Nevada, 3–7 June 1996, pp. 106–111
13 Selic, B., and Rumbaugh, J.: 'Using UML for modeling complex real-time systems'. Technical report, Rational Software, 1998
14 Selic, B., Gullekson, G., and Ward, P.: 'Real-time object-oriented modelling' (John Wiley & Sons, New York, 1994)
15 Martin, G.: 'UML for embedded systems specification and design: motivation and overview'. Proc. Design Automation and Test in Europe Conf., Paris, France, 4–8 March 2002, pp. 773–775
16 Harel, D., and Politi, M.: 'Modelling reactive systems with statecharts: the statemate approach' (McGraw-Hill, New York, 1998)
17 European Telecommunication Standards Institute (ETSI), broadband radio access network (BRAN), 'HIPERLAN type 2: data link control (DLC) layer – part 1: basic data transport functions', TS 101 761-1, November 2000
18 European Telecommunication Standard Institute (ETSI), broadband radio access network (BRAN), 'HIPERLAN type 2: data link control (DLC) layer – part 2: radio link control (RLC) sublayer', TS 101 761-2, April 2000
19 European Telecommunication Standard Institute (ETSI), broadband radio access network (BRAN), 'HIPERLAN type 2: packet based convergence layer - part 2: ethernet service specific convergence sublayer (SSCS)', TS 101 493, April 2000
20 Keutzer, K., Malik, S., Newton, A., Rabaey, J., and Sangiovanni-Vincentelli, A.: 'System-level design: orthogonalisation of concerns and platform-based design', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2000, **19**, (12), pp. 1523–1543
21 ARM Limited, 'Integrator/AP user guide', 1999–2001
22 ARM Limited, 'ARM7TDMI technical reference manual', Rev. 4.0, 2001
23 Cusani, R., Torregiani, M., Delli Priscoli, F., and Ferrari, G.: 'A novel MAC and scheduling strategy to guarantee QoS for the new-generation WIND-FLEX wireless LAN', *IEEE Wirel. Commun.*, 2002, **9**, (3), pp. 46–56
24 Rational Software Corp., 'Rational rose real time: modeling language guide', 2002
25 ARM Limited, 'AMBA specification', Rev. 2.0, 1999
26 ARM Limited, 'Integrator/LM-XCV600E + and LM-EP20K600E + user guide', 2000
27 Xilinx Inc., 'Virtex-E 1.8V field programmable gate arrays, preliminary product specification', Version 2.0, 2001
28 Kuncicky, D.: 'MATLAB programming' (Prentice Hall, New York, 2003)
29 Model Technologies Corp. (Mentor Graphics Corp.), 'Modelsim PE – simulation and verification datasheet', 2002
30 Khun-Jush, J., Schramm, P., Wachsmann, U., and Wegner, F.: 'Structure and performance of the HIPERLAN/2 physical layer'. Proc. IEEE Conf. on Vehicular Technology, Houston, TX, 16–20 May 1999, pp. 2667–2671
31 Chen, R., Sgroi, M., Martin, G., Lavagno, L., Sangiovanni-Vincentelli, A., and Rabaey, J.: 'Embedded system design using UML and platforms'. Proc. Forum on Specification and Design Languages, Marseille, France, 24–27 September 2002