# Energy-Aware System-on-Chip for 5 GHz Wireless LANs*

Labros Bisdounis[1], Spyros Blionas[1], Enrico Macii[2], Spiridon Nikolaidis[3], and Roberto Zafalon[4]

[1] INTRACOM S.A., Markopoulo Ave., GR-19002 Peania, Athens, Greece
[2] Dipartimento di Automatica e Informatica, Politecnico di Torino, I-10129 Torino, Italy
[3] Department of Physics, Aristotle University of Thessaloniki, GR-54124 Thessaloniki, Greece
[4] STMicroelectronics, Advanced System Technology, I-20041 Agrate Brianza, Milan, Italy

**Abstract.** This paper presents the realization of an energy-aware system-on-chip that implements the baseband processing as well as the medium access control and data link control functionalities of a 5 GHz wireless system. It is compliant with the HIPERLAN/2 standard, but it also covers critical functionality of the IEEE 802.11a standard. Two embedded processor cores, dedicated hardware, on-chip memory elements, as well as advanced bus architectures and peripheral inter-faces were carefully combined and optimized for the targeted application, leading to a proper trade-off of silicon area, flexibility and power consumption. A system-level low-power design methodology has been used, due to the fact that power consumption is the most critical parameter in electronic portable system design. The 17.5 million-transistor solution was implemented in a 0.18 micron CMOS process and performs baseband processing at data rates up to 54 Mbit/s, with average power consumption of about 550 mW.

## 1 Introduction

In wireless data communications there have been many standardization efforts in order to meet the increased needs of users and applications. In the 5 GHz band, there were the IEEE 802.1a [1] and the HIPERLAN/2 [2-3], both specified to provide data rates up to 54 Mbps for wireless LAN applications in indoor and outdoor environments. Both aforementioned standards operate in the same frequency band, and utilize orthogonal frequency division multiplexing (OFDM) for multicarrier transmission [4].

The purpose of this paper is to present the realization of a low-power system-on-chip that implements the baseband processing as well as the medium access control and data link control functionalities of a 5 GHz wireless LAN system. Wireless communication systems require optimization of different factors including real time performance, area, power, flexibility and time-to-market. In order to optimize the combination of the above factors, instruction-set processors, custom hardware blocks as well as low-power memory and bus interface synthesis and mapping techniques are followed, offering a good balance between flexibility and implementation efficiency. The evolving scenario has serious consequences for any SoC development to be used in wireless systems. The protocol processor (running the upper protocol's layers) is included to the implemented SoC on the contrary to previous ASIC implementations

* This work was partially supported by the IST-2000-30093 EASY project

V. Paliouras, J. Vounckx, and D. Verkest (Eds.): PATMOS 2005, LNCS 3728, pp. 166–176, 2005.

[5], [6] in which an external protocol processor should be used. Additional advantages of the implemented architecture are the adopted low-power design methodology, and the flexibility inserted by using an additional embedded processor core for controlling the baseband modem and implementing the lower-MAC processing of the HIPERLAN/2 standard, and a custom processor (MAC hardware accelerator) for implementing the lower MAC processing of the IEEE 802.11a standard [7]. During the development of the SoC, power optimization techniques were applied in the whole design range of the system, starting from the embedded software and the hardware-software mapping until the memory and bus interface synthesis [7].

The system-level design, the architectural choices of the implemented SoC, as well as the validation methodology have been presented in detail in [7-9]. In this paper we present an overview of the used design methodology, while our emphasis is on the presentation of the implementation results.

## 2 System-Level Design and SoC Architecture

The major problem in the design of a complicated system, such as a wireless LAN SoC, is the verification of the design and the early detection of design faults. For this reason a UML-based flow for the system design was followed in order to produce an executable system specification (virtual prototype) for early verification. This virtual prototype is based on a UML model, which uses the UML-RT profile [10]. The structure of the UML-based system model is presented in Figure 1.

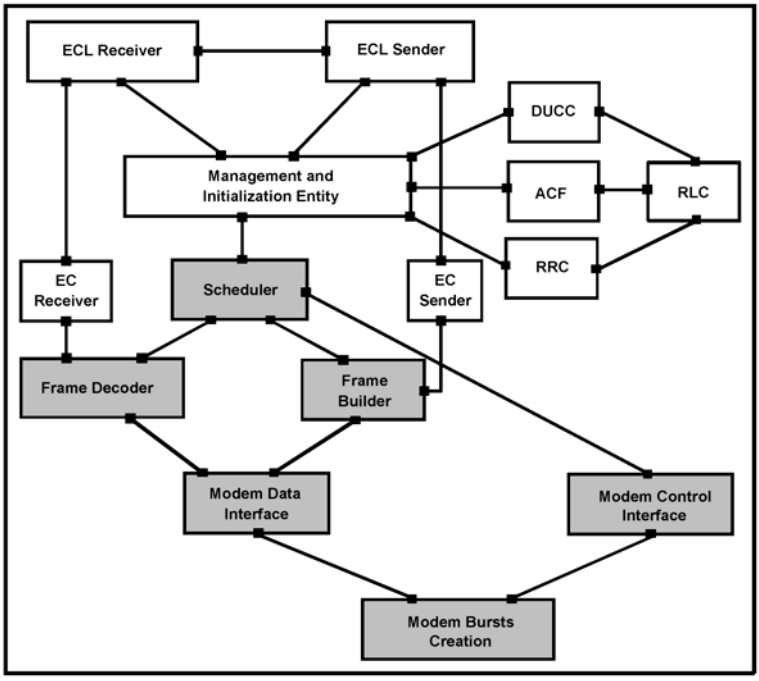


**Fig. 1.** Object structure of the HIPERLAN/2 high-level model

Within the structure of the model there is a group of three objects (modem data interface, modem control interface, modem bursts creation) that model the interface of the modem with the rest system, as well as the behavior of the modem's control parts. The remaining objects regard the upper layers of the HIPERLAN/2 protocol stack [2] and a custom implementation of the lower-MAC layer.

In order to support the hardware-software mapping of the system, a profiling and energy estimation methodology and tool were developed. The tool is based on a commercial instruction-set simulator, augmented with functional, timing and power models for peripherals and memory systems. It takes, as input, the executable specification (source code) of the target application, as well as power and timing models for various system components and it generates detailed profiling information, with aggregate performance and energy estimation for all functions in the source code. The profiling data are then used to identify the most energy and performance critical kernels in the application, in order to enable efficient mapping of the functional specification of the chip onto the target architectural template (Figure 2).

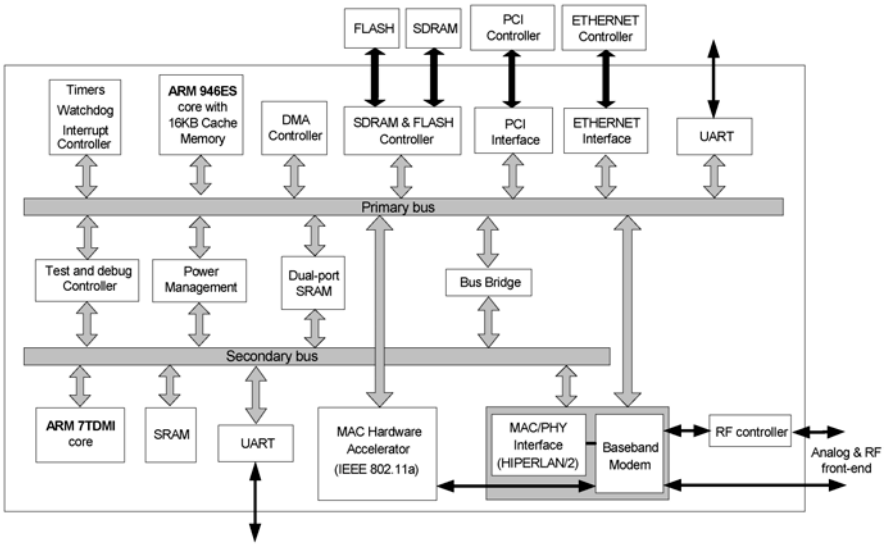


**Fig. 2.** Architectural template of the SoC

We have evaluated the impact of the mapping approach on the power consumption of the baseband modem-related circuitry, where after the profiling we migrated the HIPERLAN/2 lower-MAC functions from hardware to software, and we compared the result with that produced by using an open-source partitioning tool in which performance and hardware resources are taken into account without any power parameter. The achieved power saving was 19 %. Power estimations were taken by using the PowerChecker tool [11].

The SoC is based on a memory-mapped architecture with a primary (AMBA AHB bus) and a secondary bus (interfaced through a bus bridge). The main macrocells of the chip are: protocol processor (ARM946E-S core with 16KB Cache memory along with ARM-related blocks, PCI and Ethernet interfaces for the mobile terminal and

access point devices respectively, SDRAM and Flash controller, internal SRAMs (single and dual port), DMA controller, test and debug controller, power management unit (accounting for the special circuitry for clock gating and supply shutdown), two UART I/O serial interfaces, dedicated ARM7TDMI core (for controlling the baseband modem and support critical lower-MAC processes), MAC hardware accelerator (supporting critical lower-MAC functionality of the IEEE 802.11a standard such as encryption/decryption, fragmentation, timing control, protocol medium access) [1], dedicated block for the interface of the MAC and PHY processing elements (for the case of the HIPERLAN/2 standard), and baseband modem implementing the digital part of the physical layer of both HIPERLAN/2 and IEEE 802.11a standards [1-3].

## 3 Low-Power Design Methodology

The first stage of the applied low-power design methodology regards the preprocesssing and pruning of the executable specification in order to extract useful information and produce a pruned version of the code, which will be used in the next optimization stage that regards data transfer and storage energy optimization. At the preprocessing stage, the information concerning the algorithmic behavior was extracted, as well as the different tasks and their communication way, the used data types and the manner in which data are manipulated. During pruning some parts of the code were removed as they are consider unworthy of being optimized. In the second stage, a transformation-based methodology was applied to the pruned code guided by power and performance cost functions. The output is an optimized code in terms of memory accesses, which contributed in the reduction of the total power consumption of the system. In order to evaluate the impact of the applied software optimizations we performed an experiment, in which we estimated the power consumed by the execution of the software specification on the ARM946-ES processor. The achieved power saving in comparison with the original executable specification was 18 %.

In addition, a methodology and an instrumentation setup for measuring the instruction-level power consumption of an embedded processor were developed [12]. They are based on the measurement of the instantaneous current drawn by the processor in each clock cycle. The instrumentation setup for the accurate measurement of the processsor current includes a current sensing circuit, a digital storage oscilloscope and a PC with data processing software. After continuous monitoring and measurement of the instantaneous current of the processor, we derived instruction-level power models. Based on these models a framework was developed for the estimation of the energy consumed by the software running on the ARM processors. The framework calculates the base and inter-instruction energy costs of a given program and takes into account all other factors influencing the total power consumed by the software. Estimations and analysis that were performed using the developed framework were used for the optimization of the lower-MAC processes running on the ARM7TDMI processor.

Based on existing memory partitioning techniques [13], an automatic optimization methodology for on-chip memories was applied to the main memories of the SoC. According to the used methodology, after the mapping of the memory addresses range onto an on-chip SRAM and the analysis of its dynamic access profile, a multi-banked memory architecture is synthesized, optimally fitted to such profile [13]. The profile (obtained by instruction-level simulation) gives for each address in the range, the

number of reads and writes to the address during the execution of the target application. Assume for instance the profile of Figure 3, where a small subset of the addresses is accessed very frequently. A power-optimal partitioned memory organization is shown in the right part of Figure 3 that consists of three memory cuts and a memory selection block. The larger cuts contain the top and bottom part of the range, while 'hot' addresses are stored into a small memory. The average power in accessing the memory is decreased, because a large fraction of accesses is concentrated on a small memory, and the memory banks that are not accessed are disabled through chip select (CS). According to our system architecture (Figure 2), there are in principle two internal memory structures on which memory partitioning was applied: a single-port 16KB SRAM and a dual-port 120KB SRAM. Table 1 gives the memory components of each memory after their partition made according to the analysis of the access profile. The power savings were about 22% for the case of the single-port SRAM, and 62% for the case of the dual-port SRAM. Power estimations were taken by using the PowerChecker tool [11].

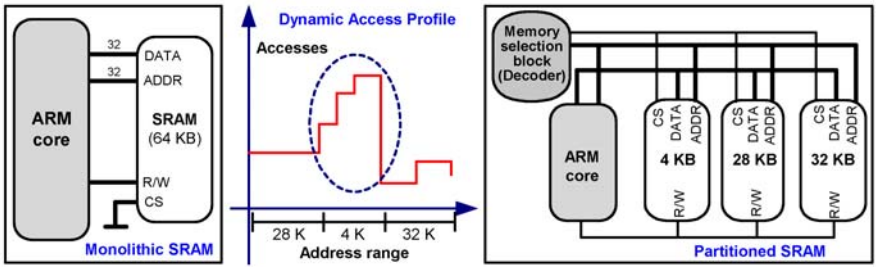


**Fig. 3.** Memory partitioning for power optimization

**Table 1.** On-chip memories partitioning

| Memory components | Components after partition |
|---|---|
| Single-port SRAM: 16 KB | Block 1: 3.4 KB<br>Block 2: 12.6 KB |
| Dual-port SRAM: 120 KB | Block 1: 1 KB<br>Block 2: 119 KB |

Another method that was used to reduce the power consumption in the SoC was to apply existing data encoding techniques on the information transmitted on the buses. These techniques consist of modifying the way the binary words are represented. In order to apply bus-encoding techniques on the secondary bus of the SoC (see Figure 2), an exploration tool has been developed, in which several of the existing encoding methods (bus invert [14], gray [15], zone [16], adaptive [17] etc.) have been imple-mented in software. Data and address traces that have been obtained by profiling of the embedded application were used as input to the exploration tool. The output of the exploration tool is the savings in terms of power-consuming transitions number. Given a set of input bus traces, regarding addresses and data, the first step we have performed was to identify the most convenient encoding scheme using our exploration tool. After the exploration regarding the address secondary bus we found that significant power savings (33%) are achieved by applying gray coding [15], while regarding the data bus we found that power savings up to 31% are achieved by ap-

plying bus-invert [14] coding. The derivation of the above savings took into account the energy consumed by the required encoders and decoders.

## 4 Embedded Software Development

The software parts of the system include high-level protocol oriented processes, low-level protocol processes and system specific hardware drivers. The different parts of the system software can be categorized into two different types: software processes and driver parts (Figure 4). The development of the software processes has been performed at a high-level of abstraction, using UML as a modelling language [10], while code generation techniques were utilised for the production of the executable code.

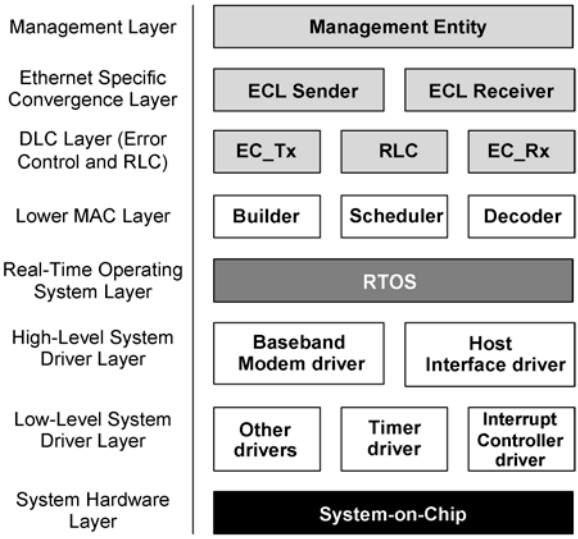


**Fig. 4.** Software architecture

Based on the results of the mapping procedure (section 2), as well as on exhaustive simulation using the UML-based high-level model of the system, a software-mapping scheme was derived and presented in the following table.

**Table 2.** Mapping of the software processes

| ARM946ES | ARM7TDMI |
|---|---|
| ECL Sender | Scheduler |
| ECL Receiver | Frame builder |
| Management entity | Frame Decoder |
| EC Transmit | Modem driver |
| EC Receive | Interrupt Controller driver |
| RLC | Timer driver |
| Host Interface driver | UART driver |
| Interrupt Controller driver | |
| Timer driver | |
| DMA driver | |

## 5 SoC Implementation and Validation

The chip is built on a dual-bus architecture (Figure 2), each bus dealing with different layers of the protocol stack. This allows each layer to have all the necessary resources for data and control transfer without having to compete for a single-bus with other layers. Furthermore, as each bus has a processor core attached, the layers become independent in what regards processing and control resources. The two buses communicate through a dual-port RAM, where large data-blocks are placed by one bus for the other to read and through a bridge, which allows the upper bus to push (or pull) small mounts of control/data information to (or from) the lower bus. Another device that connects to both buses is the baseband modem. Since we want to transfer data between layers, the modem has ports to both buses. This relieves the secondary bus from simply acting as a mediator between the modem and the upper protocol layers. The primary bus is the "protocol" bus because resources on this bus implement all the complicated classification and data transfer operations that are inherent to the protocol. The secondary bus is the "modem-control" bus. It deals with the lower layers of the protocol, acting mainly as the control and local data transfer resource for the baseband modem.

The followed low-level co-simulation procedure consisted of two main phases. The first phase exploited the advantage offered by a pure VHDL-based design and simulation environment by introducing a translator that converts the embedded core program into a set of force files for the VHDL-based simulation environment [8-9]. The second phase (FPGA-based design evaluation) [7-9] was based on the ARM Integrator platform that consists of a main motherboard implementing the system architecture, two core modules containing the required ARM cores with their peripherals, and two logic modules (hosting two XILINX Virtex E 2000 FPGAs that implement the rest logic). The average utilization of the two FPGAs was 87%.

The physical design of the chip was performed using the Magma flow [18] and contained steps such as: pads selection, IO-padring design, package selection and bonding, floorplanning, place and route, formal verification, parasitics extraction, pre/post-layout static timing analysis, post-layout simulation, physical verification, DRC, and mask preparation. Implementation data for the chip are given in the following table. Figure 5 shows the placement of the chip's macrocells and a die photograph.

**Table 3.** Chip implementation data

| | |
|---|---|
| Process technology | 0.18 μm CMOS |
| Supply voltage | 1.8 V (core), 3.3 V (I/O pads) |
| Operating frequency | 80 MHz (some modem's blocks in 40 MHz) |
| Average power consumption | 554 mW |
| Equivalent gates count | 4,400,000 |
| Transistors count | > 17,500,000 |
| Pins count | 456 (200 of them are for testing/debugging purposes) |
| Packaging | BGA 35mm x 35mm |
| Chip area | 9.408mm x 9.408mm ≈ 88.5 sq. mm |
| Core area | 8.576 mm x 8.576 mm ≈ 73.5 sq. mm |
| Area occupied by logic | 43 sq. mm |
| Area occupied by memories | 30.5 sq. mm |
| Total length of interconnections | 47 m (six metal layers) |

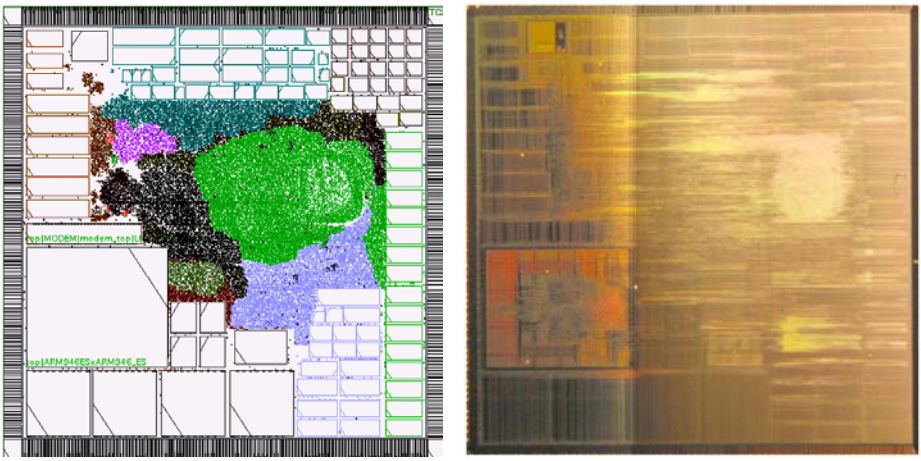

**Fig. 5.** Placement of the chip's macrocells and photograph of the die

The power consumption of the chip was estimated by using the PowerChecker tool [11]. The estimation was based on a testbench that includes an access point (AP) and a mobile terminal (MT) instance (side) of the chip that exchanging data. The total estimated power for the AP side was 460 mW, while for the MT side was 480 mW. For the embedded processors, we used the nominal power consumption (mW/MHz) for the target technology. In the following figure, we present the contribution of the main macrocells to the total chip power consumption.

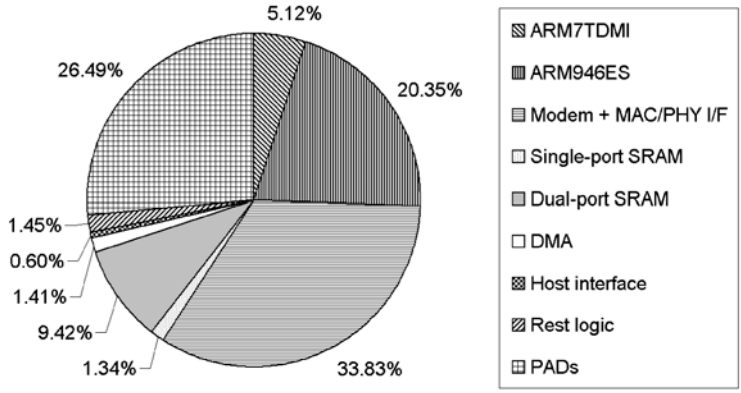


**Fig. 6.** Contribution of the system parts to the total power consumption

Power measurements were taken after the placement of the chip in two reference boards (operating as AP and MT, respectively). The average power consumption of the chip hosted by the AP board was 545 mW, while the average power consumption of the chip hasted by the MT board was 563 mW. In comparison with the power estimations given above, there was a deviation of about 18%.

In order to demonstrate and validate the SoC, we developed a system of reference boards. The system contained three boards: the main board hosting the SoC and im-

plementing the digital part of the protocol as well as the analog to digital and digital to analog conversions, the IF board and the RF board. The main board (Figure 7) contains the following major items: Ethernet controller, SDRAM memory, Flash memories, A/D and D/A conversion and clock generation circuitry, power supply circuitry, various connectors for system's interconnection and test purposes, and of course the implemented system-on-chip (SoC).


**Fig. 7.** Reference board hosting the implemented SoC

The IF board is based on a monolithic CMOS fully differential IF transceiver conver-ting an OFDM signal from 880MHz to 20MHz, while the RF board is based on a mo-nolithic SiGe RF front-end designed to transmit and receive OFDM signals in the 5GHz band. Both boards are shown in Figure 8.

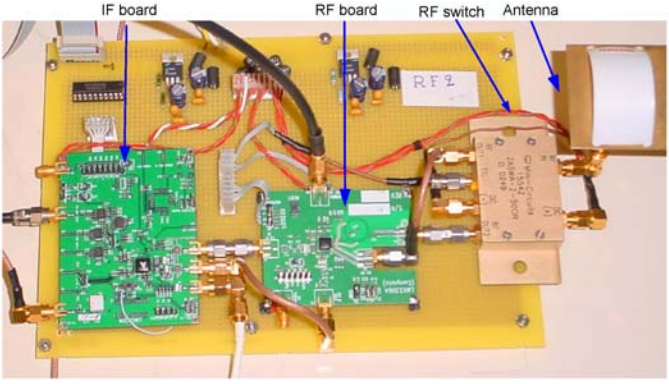


**Fig. 8.** IF and RF boards

The target area of the final demonstration setup of the EASY project includes two reference board systems (one operating as AP and a second operating as MT) exchanging Ethernet data through 5GHz RF connection. Operations such as association (between MT and AP) and connection setup, as well as network applications such as ping and FTP were tested successfully.

## 6 Conclusion

In this paper, the realization of an energy-aware system-on-chip for 5 GHz wireless LANs has been presented. The SoC implements the baseband processing as well as the medium access control and data link control functionalities of a 5 GHz wireless system. It is compliant with the HIPERLAN/2 standard, but it also covers critical functionality of the IEEE 802.11a standard. The embedded software development, the hardware implementation and the application of power optimization techniques have been described. Finally, implementation data as well as power measurements and the followed validation strategy were presented. The implemented SoC meets the functional and timing requirements of the WLAN system, and offers a power-efficient and flexible solution.

## References

1. IEEE: 'Supplement to IEEE standard for information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: WLAN Medium Access Control & Physical Layer in the 5 GHz band', IEEE Std. 802.11a, 1999.
2. ETSI BRAN: HIPERLAN Type 2: 'System overview', TR 101 683, February 2000.
3. ETSI BRAN: HIPERLAN Type 2: 'Physical (PHY) layer', TS 101 475, April 2000.
4. Van Nee, R., Prasad, R.: 'OFDM for mobile multimedia communications', Artech House, MA (1999).
5. Kneip, J., Weiss, M., Drescher, W., Aue, V., Strobel, J., Oberthur, T., Bole, M., Fettweis, G.: 'Single chip programmable baseband ASSP for 5 GHz wireless LAN applications', IEICE Trans. Electronics, vol. 85 (2002) 359–367.
6. Eberle, W., Derudder, V., Vanwijnsberghe, G., Vergara, M., Deneire, L., Van der Perre, L., Engels, M.G.E., Bolsens, I., 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, vol. 36 (2001) 1829–1838.
7. Bisdounis, L., Dre, C., Blionas, S., Metafas, D., Tatsaki, A., Ieromnimon, F, Macii, E., Rouzet, Ph., Zafalon, R., Benini, L.: 'Low-power system-on-chip architecture for wireless LANs', IEE Proc. Computers and Digital Techniques, vol. 151 (2004) 2-15.
8. Drosos, S., Bisdounis, L., Metafas, D., Blionas, S., Tatsaki, A., Papadopoulos, G.: 'Hardware-software design and validation framework for wireless LAN modems', IEE Proc. Computers and Digital Techniques, vol. 151 (2004) 173-182.
9. Drosos, S., Bisdounis, L., Metafas, D., Blionas, S., Tatsaki, A.: 'A multi-level validation methodology for wireless network applications', pp. 332-341, in Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation, edited by E. Macii, V. Paliouras, O. Koufopavlou, Lecture Notes in Computer Science Series, No. 3254, Springer-Verlag, Berlin, September 2004.
10. Martin, G.: 'UML for embedded systems specification and design: Motivation and overview', Proc. DATE, Paris, France, 4-8 March (2002) 773-775.
11. BullDAST s.r.l., "PowerChecker", User Manual, Version 4.0, 2004.
12. Nikolaidis, S., Kavvadias, N., Neofotistos, P., Kosmatopoulos, K., Laopoulos, T., Bisdounis, L.: 'In-strumentation set-up for instruction-level power modeling', pp. 71-80, in Integrated Circuit Design: Power and Timing Modeling, Optimization and Simulation, edited by B. Hochet, A.J. Acosta, M.J. Bellido, Lecture Notes in Computer Science Series, No. 2451, Springer-Verlag, Berlin, Sept. 2002.

13. Benini L., Macchiarulo, L., Macii, A., Poncino, M.: 'Layout-driven memory synthesis for embedded systems-on-chip', IEEE Trans. on VLSI Systems, vol. 10 (2002) 96-105.
14. Stan, M.R., Burleson, W.P.: 'Bus-invert coding for low-power I/O', IEEE Trans. on VLSI Systems, vol. 3 (1995) 49-58.
15. Su, C.L., Tsui, C.Y., Despain, A.M.: 'Saving power in the control path of embedded processors', IEEE Design and Test of Computers, vol. 11 (1994) 24-30.
16. Musoll, E., Lang, T., Cortadella, J.: 'Working-zone encoding for reducing the energy in microprocessor address buses', IEEE Trans. on VLSI Systems, vol. 6 (1998) 568-572.
17. Benini, L., Macii, A., Macii, E., Poncino, M., Scarsi, R.: 'Architectures and synthesis algorithms for power-efficient bus interfaces', IEEE Trans. on Computer-Aided Design, vol. 19 (2000) 969-980.
18. Magma Design Automation, "Gain-based synthesis: Speeding RTL to silicon", White paper, 2002.