Design and Implementation of a MAC Controller for the IEEE802.11 Wireless LAN

M. Iliopoulos, A. Maniatopoulos and T. Antonakopoulos

International Journal of Electronics

VOL. 88 NO. 3, 2001, pp. 271-285

INT. J. ELECTRONICS, 2001, VOL. 88, NO. 3, 271–285

Design and implementation of a MAC controller for the IEEE802.11 wireless LAN

MARIOS ILIOPOULOS†, ALEX MANIATOPOULOS† and THEODORE ANTONAKOPOULOS*‡

IEEE 802.11 wireless LANs have a tremendous market potential, as they support high data rates, work in the world-wide licence free 2.4 GHz ISM band and have high performance in terms of range and power consumption. Hence, the development of a communication processor that supports the IEEE 802.11 MAC functions has a significant potential, making very important the concept of having an ASIC 'right from the first time', in order to minimize development cost and to meet short time-to-market requirements. This paper presents the methodology of developing such a component with emphasis on the rapid prototyping approach. The chip architecture, which is based on an ARM processor core, is described in detail, focusing on the implementation of the protocol functions using custom hardware modules. Finally, the paper presents experimental results on the ASIC implementation.

1. Introduction

Wireless local area networks (LANs) are becoming very popular owing to their inherent flexibility and ease of use. The recently approved IEEE 802.11 Standard (IEEE 1997) for Wireless LANs (WLANs) is a very significant milestone in the evolution of wireless networking technology. The standard is limited in scope to the physical (PHY) and medium access control (MAC) layers of the OSI model and allows vendors to develop products as interoperable as today's wired LAN products, based on accepted industry standards. The existence of the IEEE 802.11 standard will expand the use of wireless networks in a variety of commercial and industrial applications. Target environments include indoor areas such as hospitals, residences, industries, offices and outdoor areas such as campuses, building complexes, parking lots etc.

The standard describes a PHY-independent MAC layer which may use optical or radio transmission. An optical-based PHY uses infrared light in the range of 850 nm to 950 nm and supports 1 Mbps and 2 Mbps, while the two radio PHYs use either frequency hopping (FH) spread spectrum or direct sequence (DS) spread spectrum types of radio communications. FH systems use conventional narrow-band data transmission techniques, changing their frequency periodically. The nodes hop at fixed time intervals around a spread band using different centre frequencies in a predetermined sequence. In 802.11, the FH PHY defines both 1 and 2 Mbps data rates, using 2 and 4 symbol GFSK. DS systems artificially broaden the bandwidth

Received 22 December 1999. Accepted 9 October 2000.

*Corresponding author. e-mail: antonako@ee.upatras.gr

† ATMEL – Hellas SA, Patras Science Park, Stadiou Street, Platani 26500, Patras, Greece.

‡ University of Patras, Department of Electrical Engineering and Computers Technology, 26500 Rio – Patras, Greece.

needed to transmit a signal by modulating the data stream with a spreading code. The receiver can detect error-free data even if noise exists in portions of the transmission band. The DS PHY version of 802.11 defines both 1 and 2 Mbps data rates, using differential binary phase shift keying (DBPSK) and differential quadrature phase shift keying (DQPSK).

The MAC layer of the IEEE 802.11 standard provides the services of MSDU delivery, authentication and privacy. The IEEE 802.11 MAC protocol uses the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) algorithm for delivering MAC service data units (IEEE 1997). Within a wireless system, the medium is not bounded as with a wired system. In order to control the access to the network, stations must initially establish their identity, so a type of authentication procedure is also used. Another important issue in wireless networks is privacy, which in IEEE 802.11 WLANs is implemented by using the Wired Equivalent Privacy (WEP) algorithm.

The objective of the work presented in this paper is the implementation of a communication processor capable of supporting the IEEE 802.11 MAC functions. The processor should incorporate all the necessary logic for interfacing to wireless PHYs (DS or FH), memory devices and host processors, allowing system designers to minimize the system components. Moreover, such a chip, when used in portable devices, should be targeted for low power consumption, and support of high processor speeds, in order to be able to incorporate future updates of the IEEE 802.11 protocol.

An example of a target system based on this processor (MAC chip) and direct sequence spread spectrum (DSSS) radio devices is illustrated in figure 1. The system consists of a radio chipset, the MAC processor and the memory devices. The DS configuration of the radio chipset consists of a baseband processor, a frequency synthesizer, a modem, an RF/IF converter, a power amplifier and a 2.4 GHz trans-

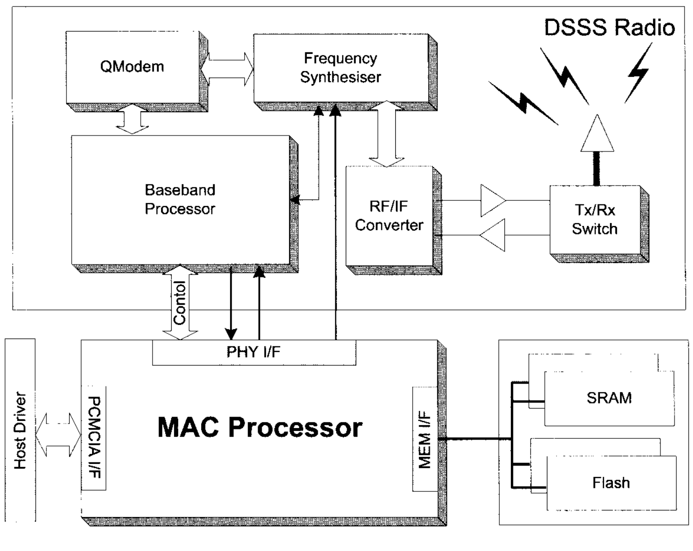

Figure 1. The system architecture.

mit/receive switch. In the FH configuration the modem and the RF/IF converter are omitted. The MAC processor implements the MAC functions and offers a glueless interface to the radio chipset, the memory devices and the PC Memory Card International Association (PCMCIA). The memory devices are used for microcode storage and temporal buffering of data packets and firmware variables.

Section 2 of this paper describes the methodology of developing such a component in terms of hardware and firmware, emphasizing on the rapid prototyping approach. The chip architecture is described in detail in § 3, focusing on the implementation of the protocol functions using custom hardware modules. Finally, § 4 presents some experimental results on the application-specific integrated circuit (ASIC) implementation.

2. System development methodology

As stated previously the main objective of this work was the implementation of a MAC processor that supports the IEEE 802.11 functions. Moreover, in order to have a viable wireless product, an equally important objective was a 'right from the first time' ASIC that could meet time-to-market requirements with minimum development cost. Meeting the above targets using the traditional method of modelling the complete system with hardware description language was not feasible, since a distributed system incorporating the MAC processor should also contain RF components, which cannot be efficiently modelled and simulated with the overall system. The access protocol complexity does not permit the evaluation of the developed microcode using only simulated hardware models and requires the deployment of a small scale network. The traditional method of developing the microcode in parallel to the hardware, and of starting the microcode testing after having the first version of the chip, increases the total development time and may result in multiple silicon runs.

The safer method of producing error-free hardware from the first run is to follow a system prototyping method using reprogrammable logic around the processor's core. This method allows the installation of an experimental network from the very beginning of the project; thus concurrent development of hardware and microcode could be performed, architectural decisions could be evaluated using experimental results and protocol interoperability tests could be performed even before the system is implemented in silicon.

The above prototyping method was used in this development and gave us the opportunity to emulate the ASIC's behaviour and to address possible architectural bottlenecks, even before starting the ASIC implementation. Based on this approach, the chip development was completed in three phases, the rapid prototyping phase, the ASIC development phase and the system testing phase.

2.1. *Rapid prototyping*

The rapid prototyping phase is divided into three subphases: system specification, emulation platform development and architecture verification, as shown in figure 2. The development of a MAC processor requires the cooperation of a hardware team that defines and implements the chip architecture and of a firmware team which develops the microcode that implements the MAC functions based on the specific hardware architecture.

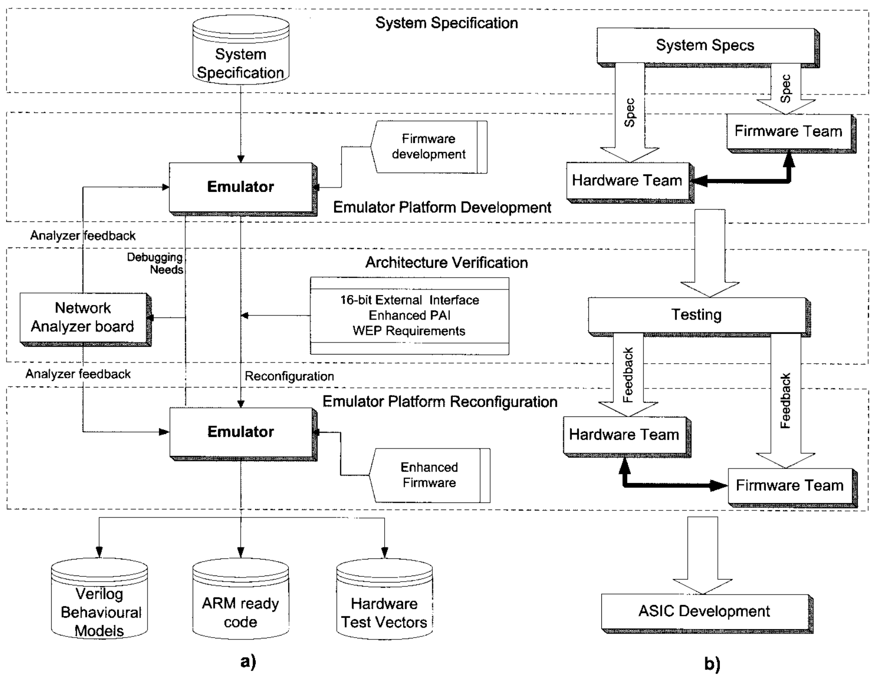

Figure 2. The rapid prototyping approach.

During system specification, the basic architectural decisions, such as custom hardware solution versus processor based solution, processor core selection etc., were considered. The specification was given to the hardware development team which was responsible for producing a prototype system that should implement the specified architecture but also be flexible (using field programmable gate arrays (FPGAs) as basic building blocks), in order to incorporate future enhancements. Using these system specifications, the firmware team worked in parallel with the hardware team to produce microcode executable in the prototyped system. The prototype (firmware/software) was given to the hardware team that tested and debugged the hardware modules and also addressed some architectural bottlenecks. The experience gained during system evaluation was used by the hardware and firmware teams to reconfigure the hardware platform and to produce an updated version of the system, on which the architecture of the final ASIC was based. The emulator board was used as the platform for the development of the microcode, while the ASIC was in design and fabrication, giving the opportunity to have an error-free code when the ASIC came out of the fabrication.

The use of the above procedure had some very important advantages as opposed to traditional ASIC targeted design. The hardware modules were tested extensively using real-time conditions and the basic hardware functions were debugged, reducing the risk of ASIC redesign. The MAC functions, implemented in firmware, were tested in the emulation hardware, and time-critical functions were addressed and transferred into hardware before the ASIC development phase was started. All teams (hardware, firmware) worked in parallel, thus minimizing the total development time significantly. More details on the rapid prototyping approach are given below.

2.1.1. *System specification.* System specification was the first step to the system development. It is considered as the most important step since many architectural parameters are defined at this stage.

The first and most important architectural decision was the selection between custom hardware and a processor-based solution. We decided that the processor solution was more suitable for this system because it requires less development time and is more flexible, using microcode for implementing some MAC functions. This solution may also support future protocol updates and other similar MAC protocols. The second issue was the selection of the processor core to be integrated into the MAC chip. The ARM processor was considered to be one of the best solutions, owing to its 32-bit powerful architecture and low power consumption. The final step in system specification was to address the major architectural blocks according to system objectives. These blocks were the memory controller, the physical attachment interface (PAI), the PCMCIA interface, and various supporting peripherals (e.g. interrupt controller, timers etc.).

The memory controller should interface the ARM processor with external memory devices. It should be configurable in order to support different memory devices and to offer compatibility with the various access modes of ARM. The physical attachment interface PAI should interface to the physical layer device, implement all bit-serial tasks (such as CRC checking) and offer a glueless programming interface to baseband and synthesizer devices. It should incorporate FIFOs for temporal buffering of network data and a DMA engine for transferring blocks of data to and from the external memory without processor intervention. The PCMCIA block should offer a glueless interface to the host driver. The peripherals should contain all the processor support functions such as timers, interrupt handling etc.

In the initial system approach, the external bus was chosen to be 8-bit in order to minimize the final pin count and the PCMCIA to be 8-bit, based on the assumption that data exchange does not exceed 2 Mbps. All hardware modules were chosen to satisfy the AMBA specification (Advanced RISC Machines Ltd 1996b).

2.1.2. *The emulator platform development.* The second step of the prototyping phase was the development of a generic emulator board for ARM based architectures. The emulator board contained an ARM processor board with a wrapper that produces the ARM synchronous bus (ASB) signals. The emulator's architecture is shown in figure 3. The memory devices contained in the system consist of SRAM and Flash for microcode development and temporal storage of variables and network data. The emulator contained all necessary interfaces to be attached to the PCMCIA bus and to the physical layer implementation.

Field programmable gate arrays (FPGAs) were used for implementing the architectural blocks defined in the system specification and offer the flexibility for system update and reconfiguration. More specifically, six FPGAs were used to implement the MAC functions, four of them connected to the ASB bus and two for implementing the peripheral devices. The ASB-attached FPGAs were used to implement the memory controller, the PCMCIA interface, the physical attachment interface, and the ASB-to-APB bridge.

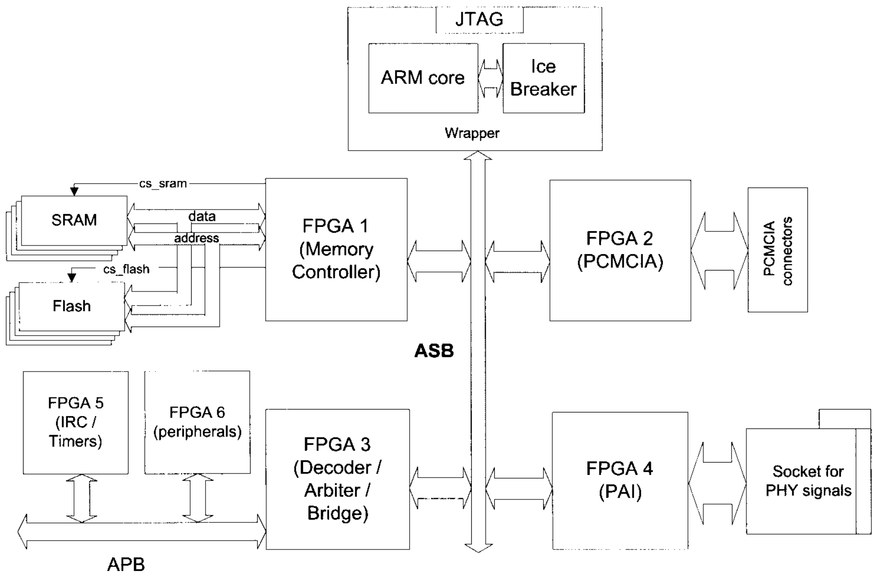

Figure 3. The emulator architecture.

2.1.3. *Architecture verification.* After the MAC processor modules had been implemented into FPGAs, the emulator board was used for firmware development. The first aim of the code development was to debug most of the custom hardware functions, testing each of the hardware modules separately. After debugging the basic functions, the firmware development targeted the IEEE 802.11 code in order to evaluate the architecture and to verify its capability to implement the IEEE 802.11 protocol. At the same time a device driver was developed in order to test the PCMCIA interface. The ARM Software Development Toolkit (ARM SDT) was the main software development platform used throughout all firmware development steps.

During the verification phase, several architectural bottlenecks were addressed, such as the 8-bit memory bus, which delayed the programme execution of the ARM processor, and the need for a fast hardware encryption/decryption module. These two bottlenecks prevented the microcode from achieving several protocol times. The increase of the external memory bus to 16 bits almost duplicated the processor's power, while the hardware encryption module released the CPU from time consuming bit-serial processing. Another enhancement was the integration of the protocol defined 64-bit timing synchronization function (TSF) timer into hardware which allowed a more strict control of time dependent functions, such as beaconing and power saving, and gave the capability to automate transmit/receive through TSF based mechanisms.

The new architectural considerations (e.g. 16-bit bus, encryption FPGA) were incorporated in the emulator board together with newer versions of firmware with enhanced functionality. The emulator was reconfigured using its flexible building blocks in order to incorporate the encryption/decryption engine (FPGA 1), an option for 8/16-bit memory bus (FPGA 1), support of the PCMCIA/ISA bus

(FPGA 2), 64-bit TSF and TSF based mechanisms (FPGA 4). These new features proved that the emulator board could be used for rapid prototyping of custom ASICs based on the ARM CPU core and not only for the MAC processor development. A photograph of the emulator board is given in figure 4.

2.1.4. *MAC-layer analyser development.* During the initial system debugging phase, the need for an IEEE 802.11 analyser became apparent, so a simple but powerful MAC layer protocol analyser was developed. The analyser was used for monitoring transmitted frames in order to check whether they conformed to the IEEE 802.11 standard specifications. The analyser consists of a PHY device and an FPGA that implements all the glue logic for directly attaching to the ISA bus. Using a device driver, the analyser was capable of receiving and storing transmitted frames in order to further debug the emulator hardware. Some of the functions implemented in the analyser were decoding and displaying of headers, displaying payloads, calculating interarrival times and gathering/displaying statistics.

At the end of the rapid prototyping phase the behavioural models of the MAC processor modules were fully debugged and tested, the microcode was developed and optimized for supporting IEEE 802.11 requirements and the hardware test vectors, used to debug the FPGA modules, were produced.

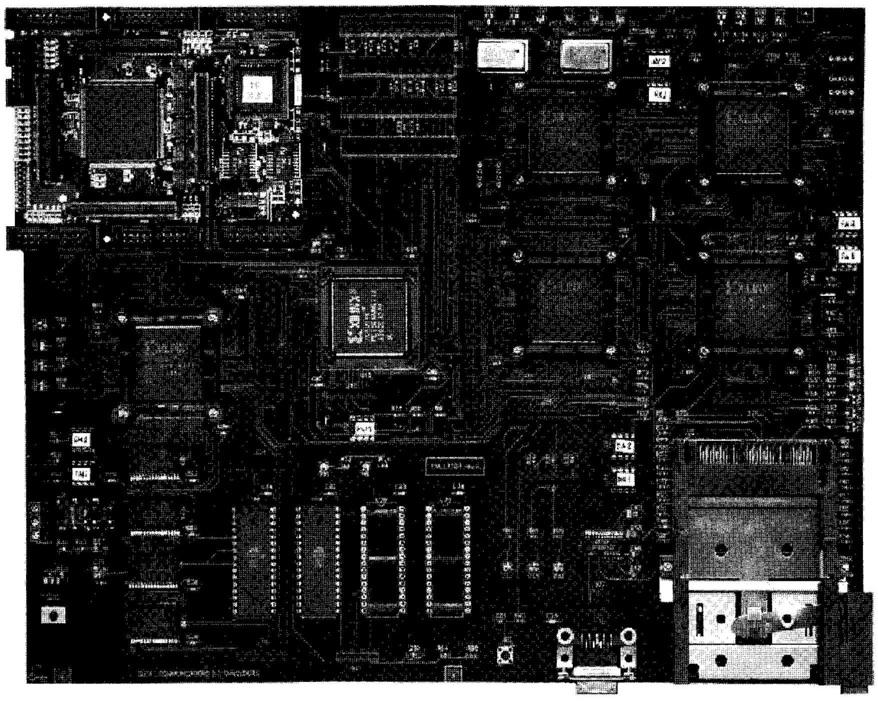

Figure 4. The emulator board.

2.2. *ASIC development*

Figure 5 illustrates the various steps of the ASIC development. The first step in this process was the integration of several behavioural models into a hierarchical view of the chip. The behaviour of the overall system was simulated using the functional test vectors produced during the rapid prototyping phase in order to test the FPGA modules. The functional emulation of the system also incorporated programme execution in the ARM processor by using the microcode developed to test the hardware modules in the emulator board. Logic errors were addressed and the required changes of the behavioural models were performed.

The next step in the ASIC development process was the gate level synthesis of the Verilog behavioural models using ES2 libraries and Synopsys synthesis tools. An important issue in this phase was the generation of the constraint files that should be given as input to the synthesis tools in order to produce efficient structural code. The constraints included specification of the maximum clock frequency, input/output delay requirements etc. The output of the synthesis step was a structural system view which contained information about the gate delays. System timing simulation gave information on critical paths and defined the maximum clock frequencies. During timing simulation, the addressed errors required changing of the initial behavioural models, re-synthesizing and timing verification.

Following the timing verification, the netlist/SDF was generated. The SDF files contained information about gate fanout, wire loads, transmission and propagation delay estimations etc. in order to track down possible timing violations. After SDF

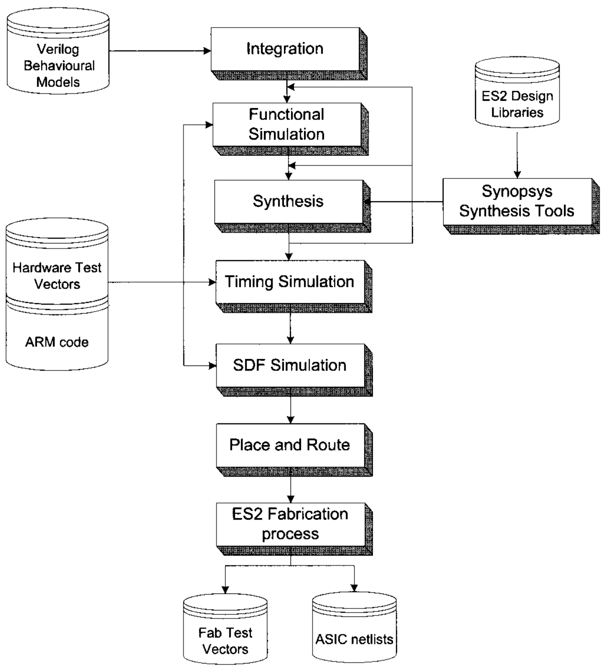

Figure 5. The ASIC development process.

generation, resimulation of the design followed, using minimum and maximum timing conditions. Timing problems that were addressed during this phase resulted in slight modifications of the behavioural/structural models and repetition of the same procedure. This step was finalized when simulations of min/max conditions produced the same results.

The 'Place and Route' process produced the final netlists that contained topological information of the cells. Clock trees and reset trees were also generated in this step. The Place and Route step produced enhanced SDF information, hence new simulation steps were needed.

The fabrication process was the final step of the ASIC development phase and produced the files needed by the fabrication to develop the masks. During this step, fabrication test vectors were generated from the functional test vectors which took into account the requirements of the available tester.

The masks produced in the final step were sent to the fabrication to proceed to ASIC manufacturing. After production of the wafer, the parts were tested using testers that applied the appropriate test patterns to each chip. After cutting, the 'good' parts of the wafer were packaged and tested using the functional test vectors. The packaged 'error-free' parts were used for real-time test/verification. The test procedure used for completing the MAC processor development is presented in § 4.

3. Chip architecture

The MAC processor architecture was finalized at the end of the rapid prototyping phase and is illustrated in figure 6. The MAC processor architecture was developed around an ARM core (Advanced RISC Machines Ltd 1996a) and its AMBA architecture (Advanced RISC Machines Ltd 1996b). The ARM processor controls all other modules through slave interfaces by using a centralized decoder. The DMA/master blocks (PAI, WEP, PCMCIA) communicate through common memory, which is read/written by these blocks, allowing data transfer without processor intervention. The bus is shared by the requesting masters through a centralized arbiter. The modules communicate through interrupt mechanisms that are controlled by the interrupt controller, which prioritizes interrupts using the fast and normal interrupt request lines of the ARM core. The interrupt controller and the timers module are connected to the ARM peripheral bus (APB) through the bridge module. The major architectural blocks of the MAC processor chip are the ARM7TDMI core with the AMBA wrapper, the physical attachment interface (PAI), the PCMCIA interface, the wireless equivalent privacy (WEP), the external memory interface, the AMBA bridge and the AMBA supporting peripherals. The ARM core implements all the MAC functions, such as frame formatting, DCF, fragmentation, reassembly, RTS/ACK/CTS. The AMBA wrapper turns the ARM core signals into AMBA signals according to the AMBA specifications.

The physical attachment interface (PAI) module performs all the necessary functions for interfacing with Wireless PHYs and has been designed to automatically handle many time-critical physical network management tasks for efficient 802.11 MAC protocol implementation. PAI contains 64-byte transmit and receive FIFOs and is capable of dynamic DMA operations. PAI also includes a 64-bit time synchronization function (TSF) counter, as specified by the IEEE 802.11 protocol. The PAI module (shown in figure 7) consists of the receive/transmit control state machines, the receive/transmit FIFOs with the respective DMA machines, the

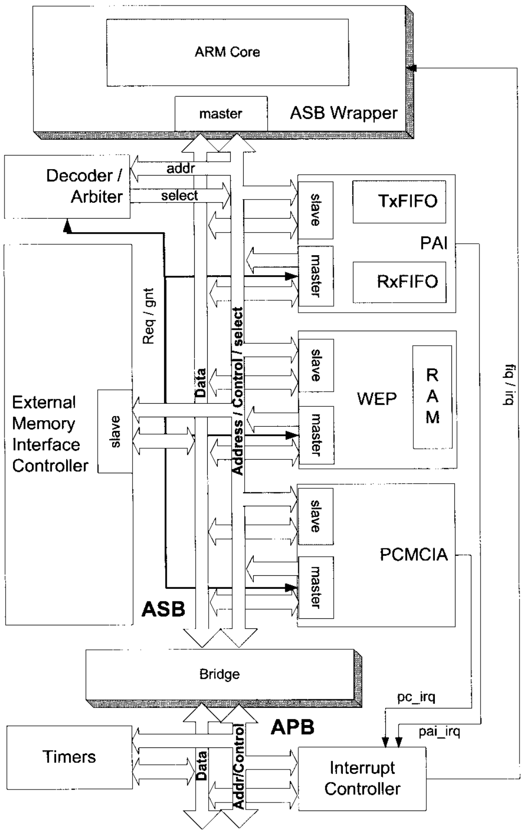

Figure 6. The MAC processor architecture.

CRC-32 calculation engine and the control/status registers. The receive/transmit control state machines control all PAI submodules, using configuration information from AMBA slave registers, and keep status information about network events. The transmit/receive DMA machines are programmed to transfer network bytes stored in FIFOs to and from the memory. The 32-bit receive and transmit bit-serial CRC engines are used for error checking data. Finally, the AMBA slave control/status registers are used to pass control information to the state machines or to read status information from the network. The TSF register is a special control register that synchronizes MAC functions and network events and is implemented as a 64-bit counter controlled by the PAI.

The PCMCIA module performs all the necessary functions for communicating with a host processor. The PCMCIA module is implemented as an AMBA master module, in order to give to the host the capability to access the MAC processor memory space. It also contains all the appropriate logic to implement the PCMCIA 2.1/JEIDA 4.2 compatible plug and play standard (PCMCIA/JEIDA, 1995). It contains a host interface part that implements all the PCMCIA functions and an AMBA interface part that implements the AMBA master/slave state machines. The

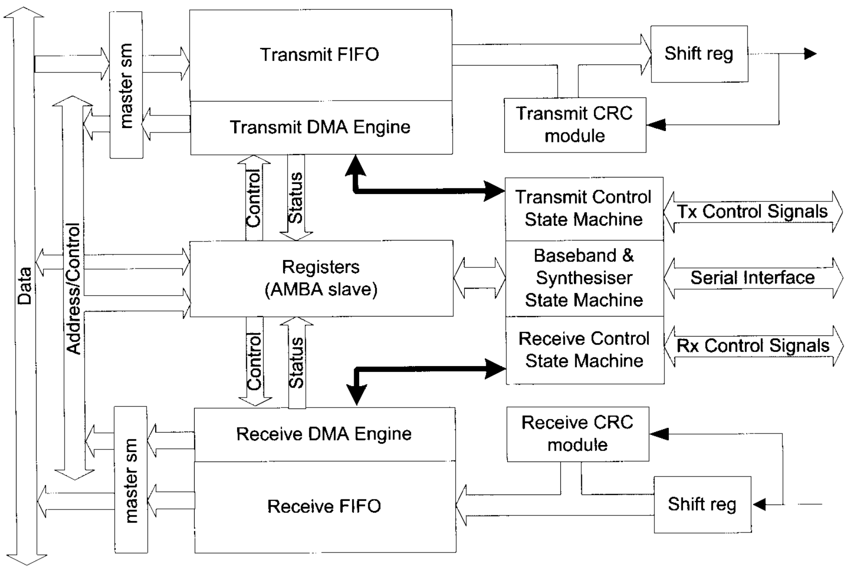

Figure 7. Physical attachment interface.

two parts communicate through interrupt and registers that are read only by the host and write only by the ARM, and vice-versa.

The wireless equivalent privacy (WEP) module implements the RC4 algorithm for encryption, decryption and integrity check of the data. The WEP module is programmed by the microcode to encrypt/decrypt a block of data in the memory. A more detailed diagram of the WEP module is illustrated in figure 8 (Iliopoulos and Antonakopoulos 1999). WEP consists of an RC4 engine that produces pseudorandom numbers, a local SRAM that stores these numbers and a read/write memory state machine which reads a block of data from memory. This state machine XORs the data with random numbers and stores them back to the main memory. Finally, WEP includes an integrity check (ICV) generation engine, which implements parallel CRC-32 generation according to the $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ generating polynomial.

The external memory interface controller, implemented as an AMBA bus slave module, performs all the necessary functions for interfacing with external memory devices supporting all transfer modes of the ARM core and using programmable wait states. The AMBA bridge module contains all the glue logic for connecting slow peripherals to the AMBA bus. It works at 1/3 of the speed of the AMBA bus clock and generates all the necessary peripheral bus signals according to the Advanced Peripheral Bus (APB) specifications (Advanced RISC Machines Ltd 1996b).

The AMBA slow peripherals are the interrupt controller (IRC) and the timers (TIM). The IRC drives the fast interrupt request and the interrupt request lines of the ARM core. It accepts all the interrupt request signals from the other modules and selects the appropriate interrupt lines according to a fixed priority scheme. The TIM module is used for MAC specific functions that require accurate time calculations. It contains two 32-bit counters with independent programmable prescaling and measures time with up to 50 ns accuracy.

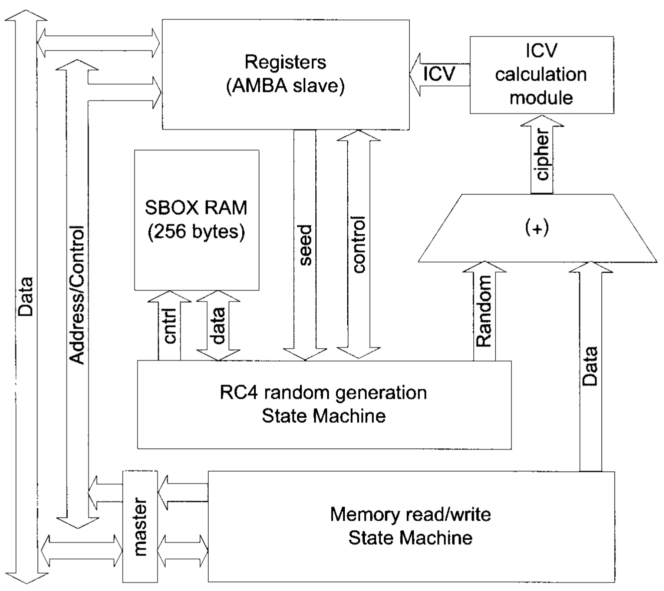

Figure 8. Wired equivalent privacy module.

The above architecture was evaluated in the emulator board and was used as the base for the ASIC development presented in § 2. The emulator platform allowed the firmware team to develop the microcode for IEEE 802.11 functions while the ASIC was in the fabrication process. The complete IEEE 802.11 microcode was almost ready before the ASIC became available. After completing all ASIC tests and optimizing the microcode, 20 Kbytes of program were required for implementing the distributed coordination function (DCF) of the IEEE 802.11 protocol.

4. ASIC testing and experimental results

As stated in § 2, the architectural modules were initially implemented using FPGAs. The family XC4000 of XILINX was considered best in this case, since it contains a variety of FPGAs in different speeds and sizes for the same physical package and pin allocation. For the emulator FPGAs, the parts that could implement more than 10k equivalent gates and contain internal memories for implementation of FIFOs and SRAM devices were selected. Table 1 illustrates the implementation of the hardware modules in the emulator's FPGAs.

The ASIC was implemented in ES2 0.5 μm technology and packaged in a 144-pin plastic thin quad flat package (TQFP) suitable for PCMCIA applications. The percentage of silicon area required by each block is shown in column 3 of table 1, while figure 9 illustrates the floorplan of the chip. The macrocells used (ARM core, SRAM, DualPort RAM) are shown with different shading.

In order to demonstrate the functionality of the chip in a real environment, a demonstrator board was constructed. The demonstrator board, illustrated in figure 10, is a PCMCIA card that consists of the MAC processor chip, Flash/SRAM devices, and a PCMCIA type socket for interfacing with the Harris MAC-less

Hardware module	No. of CLBs/ % of FPGA used[a]	% of silicon area
PAI	690 CLBs/88%	20%
WEP	315 CLBs/40%	10%
PCMCIA	221 CLBs/28%	7%
MEM_CNTL	118 CLBs/15%	5%
Timers + IRC	310 CLBs/39%	10%
ARM + wrapper	ARM board	28%
Memories	FPGA internal	16%
Test logic	–	4%
Total	1654 CLBs	100%

[a]For Xilinx XC4020E that contains 20 000 equivalent gates.

Table 1. FPGA/ASIC implementation.

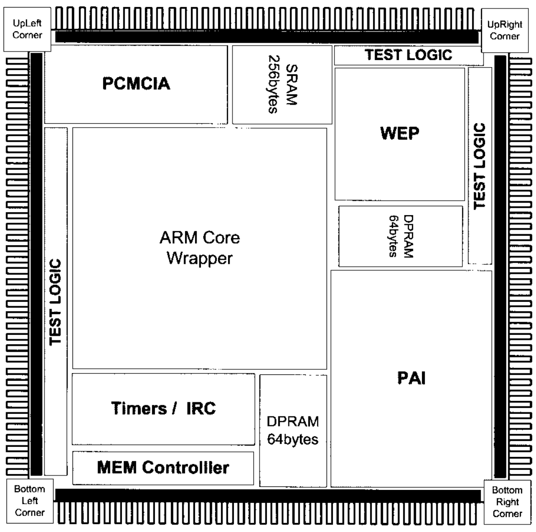

Figure 9. The chip floorplan.

board that uses the PRISM^{TM} chipset (Harris Semiconductor 1997) provided by Harris Semiconductor. Using the demonstrator board, the MAC processor ASIC was tested and proved to be fully functional. The test-bench used consisted of three stations with PCMCIA slots using the demonstrator boards and a fourth station using the analyser board for capturing and monitoring the packets. The system was tested on various network applications such as ftp transfers, telnet and web browsing.

One of the tests performed targeted on measuring the chip's power consumption. The power consumption measured for 20 MHz clock speed is illustrated in table 2. According to this table the MAC processor requires a typical power of 185 mW, which is up to 56% less than other MAC implementations as in AMD's Am76C930 Wireless LAN Medium Access Controller shown in table 3 (Advanced Micro

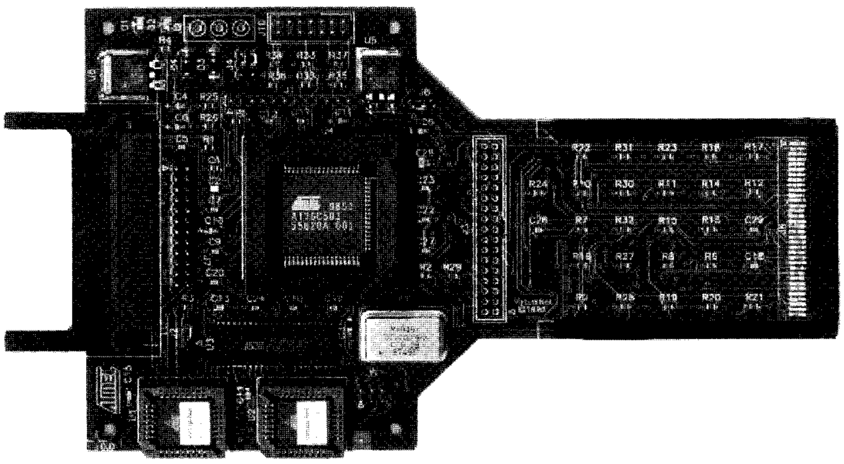

Figure 10. Demonstrator board.

Symbol	Parameter	Value	Unit
	Dual supply (5 V I/O supply; 3.3 V core supply)		
ICC	Core supply current	38	mA
IDD	I/O supply current	12	mA
PW	Total power consumption	185	mW
	Single supply (3.3 V core and I/O supply)		
ICC	Supply current	48	mA
PW	Total power consumption	158	mW

Table 2. Power consumption at 20 MHz internal clock.

Symbol	Parameter	Value	Unit
	5 V supply		
ICC	Supply current	85	mA
PW	Total power consumption	425	mW
	3.3 V supply		
ICC	Supply current	60	mA
PW	Total power consumption	198	mW

Table 3. Am76C930 Power consumption at 20 MHz internal clock.

Devices 1995), while offering more CPU power (MIPS/MHz) at the same clock speed.

5. Conclusions

The methodology described in this paper gave us the opportunity to produce an error free ASIC at the first time and to minimize development time. This was achieved by extensively testing all hardware modules on an emulation platform using real-time conditions, thus minimizing the risk of errors in the final ASIC.

Moreover, the MAC functions, implemented in microcode, were tested in emulation hardware and the time-critical processes were addressed at the initial development phase and transferred into hardware.

Future work in this field will target the new 802.11a (IEEE 1999a) and 802.11b (IEEE 1999b) higher data rate standards, using the same development platform. Work has been done also in the area of access points with architectures able to support ATM/Ethernet to Wireless bridges in a single chip (Iliopoulos *et al.* 1998).

Acknowledgements

This work was financially supported by the European Union in the framework of the OMI-ESPRIT 'VNET – Design, Implementation and Demonstration of an Adapter card for Wireless Local Area Networks' Project No. 23631.

References

Advanced Micro Devices, 1995, Publication 20183, Rev. A, Am79C930, Pcnet^{TM}—Mobile Single Chip Wireless LAN Media Access Controller.

Advanced RISC Machines Ltd., 1996a, No: DDI 0100B, ARM Architecture Reference Manual (Prentice Hall).

Advanced RISC Machines Ltd., 1996b, No: DVI 0010A, Introduction to AMBA.

Harris Semiconductor, 1997, No. AN9624.3, PRISM 2.4 GHz Direct Sequence Spread Spectrum Wireless LAN.

IEEE, 1997, Std 802.11-1997: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification.

IEEE, 1999a, Std 802.11a/D5.0, Draft Supplement to IEEE Std 802.11-1997: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification. High Speed Physical Layer in the 5 GHz band.

IEEE, 1999b, Std 802.11b/D5.0, Draft Supplement to IEEE Std 802.11-1997: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification. Higher Speed Physical Extension in the 2.4 GHz band.

Iliopoulos, M., and Antonakopoulos, T., 1999, Hardware implementation of the Wired LAN Equivalent Privacy (WEP) in 802.11 Wireless LAN. *IMACS/IEEE Circuits, Systems, Communications & Computers (CSCC'99) Conference,* Athens, Greece.

Iliopoulos, M., Maniatopoulos, A., and Antonakopoulos, T., 1998, Bridge-on-a-chip: Inter-networking ATM with the IEEE 802.11 Wireless LAN. *Networks & Optical Communications (NOC'98) Conference*, Manchester UK.

PCMCIA/JEIDA, 1995, No: 0595-02-1000, PCMCIA Card Standard, Electrical Specification, Vol. 2.