Reprint

A Methodology for Implementing Medium Access Protocols Using a General Parameterized Architecture

M. Iliopoulos and T. Antonakopoulos

The 11th IEEE International Workshop on Rapid System Prototyping – RSP'2000

PARIS, JUNE 2000

A Methodology of Implementing Medium Access Protocols Using a General Parameterized Architecture

Marios Iliopoulos, Theodore Antonakopoulos
University of Patras, Dept. of Electrical Engineering and Computers Technology
26500 Rio - Patras, Greece
mariosi@atmel.gr, antonako@ee.upatras.gr

Abstract

The aim of rapid development of communication systems is to automate the process of transforming the high level description of a protocol into hardware and software that would implement the actual system. This paper describes a methodology used to implement medium access protocols based on a microprocessor core and a general parameterized architecture which contains configurable hardware blocks that can be customized according to the protocol needs (mainly for frame based networks), thus reducing the time and effort needed to develop an embedded communication system.

1. Introduction

The trend in current network design is to combine flexibility and programmability with high performance, low power consumption and low cost. As shown in Figure 1, these requirements contradict, since systems based on microprocessors and Digital Signal Processors (DSPs) may offer flexibility and programmability, but they require more power, are less efficient and usually have increased cost. On the other hand, implementations that use dedicated logic, which is a characteristic of Application Specific Integrated Circuits (ASICs), are less flexible than any other solution, since they are targeted only for one application.

From the designer's point of view, systems that are based on programmable cores, require less development and testing time than systems which contain dedicated logic developed from scratch, since a lot of time is consumed for debugging until the system reaches a stable state. Another solution is to use a microprossesor supported by reconfigurable hardware (such as FPGAs or CPLDs) during prototyping to increase the efficiency and performance of the microprocessor core.

In this paper we consider the last solution of system prototyping where the microprocessor is supported by dedicated hardware which is produced automatically from a general parameterized architecture. The architecture is based on a library of parametric blocks that are extracted from the study of various access protocols and can be configured to implement different Medium Access Control functions. These blocks are interconnected in a flexible way in order to be adapted to the requirements of different protocols. This approach has the advantage of rapid system development, while preserving the cost, power consumption and efficiency requirements of various networks.

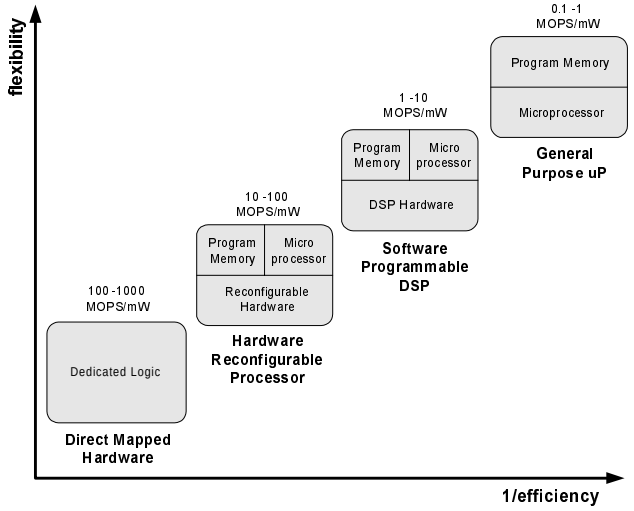

Figure 1. Architectural choices

Section 2 of this paper describes the common functions, which are met in different MAC protocols. Section 3 introduces the general network architecture and the system architecture for implementing MAC processors, while Section 4 describes the application of the general network architecture to the IEEE 802.11 MAC protocol.

2. Access Protocols Common Functions

The study of various packet based Medium Access protocols such as IEEE802.11 [1] and Bluetooth [2], that represent two modern wireless MAC standards, and IEEE 802.3 [3] the most common MAC protocol for wired networks, shows that there are many similar functions in all access protocols. These functions can be categorized as bit-serial functions that process the serial bit-stream, parallel functions that process the parallel data, event processing functions that process the network events, and control functions that synchronize all the above blocks and consist of control registers and state machines.

The *bit serial functions* consist of two main groups of functions, functions that change the serial bit-stream and functions that do not alter the bit-stream information but extract results from its content. The functions that change the serial bit-stream, like scrambling, whitening, encryption/decryption are usually cascaded. For example, in Bluetooth, each of the serial functions *Whitening*, *Forward Error Correction (FEC)* and *Encrypt/Decrypt*, has as input the output of the previous level as illustrated in Figure 2.

On the other hand, functions that do not change the serial bit-stream, like CRC32, access code detection, etc. usually work in parallel with other bit-serial functions in order to either check a block of data (like CRC-32) or produce control information (such as start of reception, synchronization pattern detection etc.).

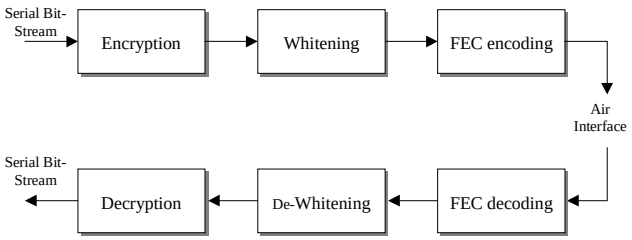

Figure 2. Cascaded bit serial functions

Like the bit-serial functions, the *parallel functions* can either change the content of the parallel data, e.g. parallel XOR of raw data with a pseudorandom number for encryption/decryption of data, (e.g. the WEP algorithm in IEEE 802.11 - Figure 3) or add data in certain positions in the packet format, such as the station address in the source address field in IEEE 802.3 protocol or the integrity check value (ICV) of the encrypted packet in IEEE 802.11 protocol. Another parallel function is to compare the contents of the data with predefined values in order to offer control information to state machines. For example, the decoding of the address field of the packet in IEEE 802.11 and 802.3 protocols gives the information for unicast, broadcast, or multicast packets in the receive direction.

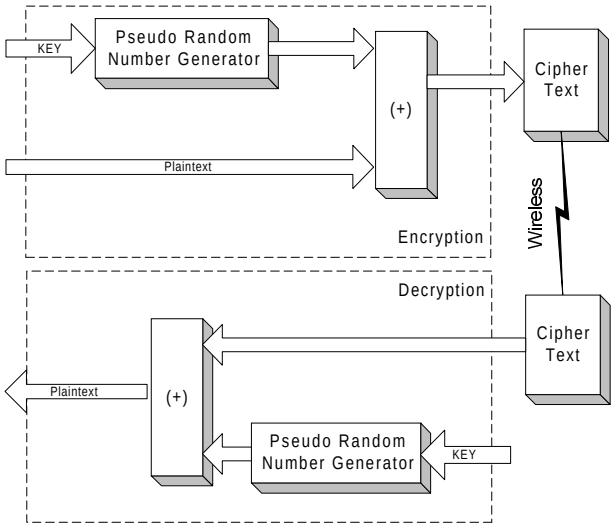

Figure 3. Parallel Encryption/Decryption process

In all Medium Access Controllers there are mechanisms that recognize events coming either from the network side (e.g. positive edge on Start of Frame, negative edge on Packet Done etc.) or from control registers, (e.g. event on a register bit, a counter overflow etc.). These mechanisms comprise the *events processing* section. The events processing section passes control information to the state machines section. For example an *'end of receive packet'* event will trigger the back-off calculation algorithm in IEEE 802.11 and 802.3 MAC protocols.

The *control functions* use registers that carry control information which is written by the microprocessor and read by the rest of the modules, status information which is updated by the modules and read by the microprocessor, or timing/statistics information which is updated periodically depending on events (timers). This information is used by the *state machines* that monitor and control the rest of the blocks, for both transmit and receive directions. The state machines process control information, synchronize events, and perform protocol specific functions. They work either synchronously, using the system clock or the network clock, or asynchronously triggered by events. A special case is the *DMA state machine*, which is an optional block that gives the capability to generate the address and control signals for transferring the data from/to FIFOs to/from the memory without microprocessor intervention. If this block is omitted then the microprocessor is responsible for data transfers.

In all access protocols FIFOs are required in both directions in order to isolate the network timing from the node's internal timing and to generate constant data transmission and reception. Most of the packet based protocols use the above blocks according to the flow illustrated in Figure 4.

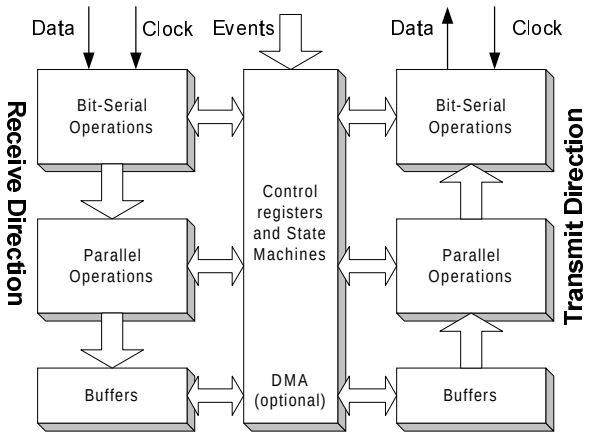

Figure 4. General Architecture Block Diagram

According to this figure, the received serial data are passed through the bit-serial and parallel operations before they are stored into buffers and processed by the upper network layers. The whole process is controlled by the state machines block which transacts with the above functions and the events coming from the network. Similarly, in the transmit direction, the data coming from the buffers are transformed through parallel and bit-serial operations into a bitstream, which is transmitted over the network.

3. The General Network Architecture

The blocks described in the previous section are combined into a general architecture that is based on the flow of Figure 4 and is capable of supporting Medium Access processing of most of the packet based networks. This architecture contains parametric blocks that can be tailored to MAC protocol needs and are interconnected through flexible interfaces.

There are two main blocks in this architecture, the Receiver section which contains all the receive related functions (Figure 5), and the Transmitter section that contains all the transmit related functions (Figure 6). The control section contains all the control registers that are programmed/read by the microprocessor through a separate control interface. The control interface can be a custom microprocessor interface, or a standard bus. The data movement from/to the memory is accomplished through a dedicated path, either transparently without processor intervention by using a DMA engine, or with processor read/writes where the DMA engine can be omitted. Each of the transmit/receive section contains the blocks described in section 2 in a flexible and parameterizable way.

The bit-serial functions block contains an array of bit-serial functions that are interconnected in such a way that each of them can work cascaded or in parallel with the others through configurable interconnections. In the receive

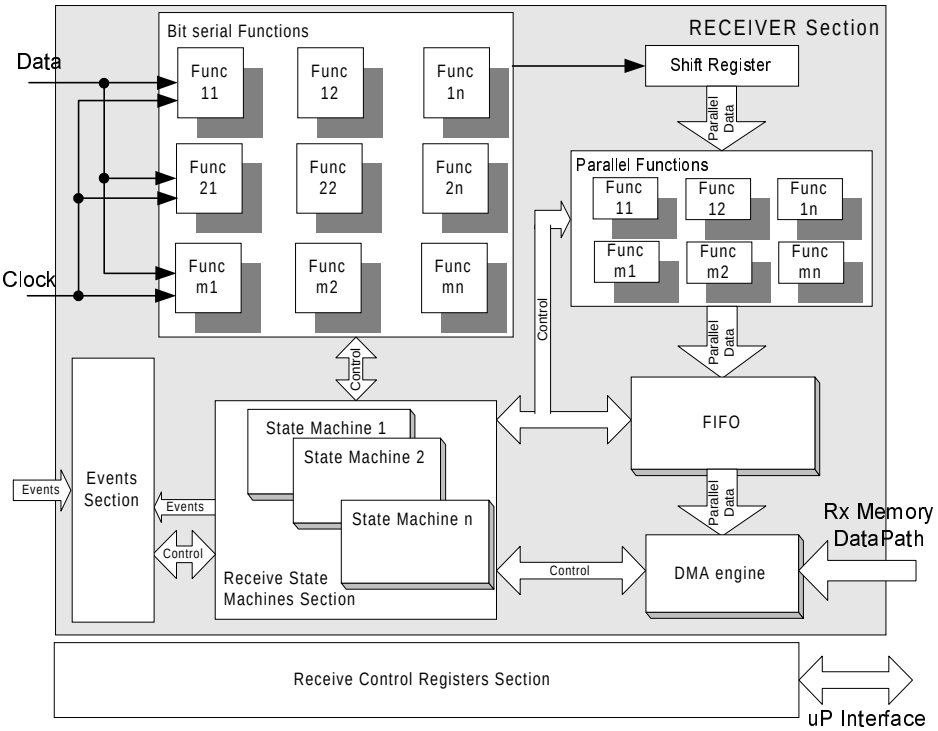

Figure 5. General Network Architecture-Receiver

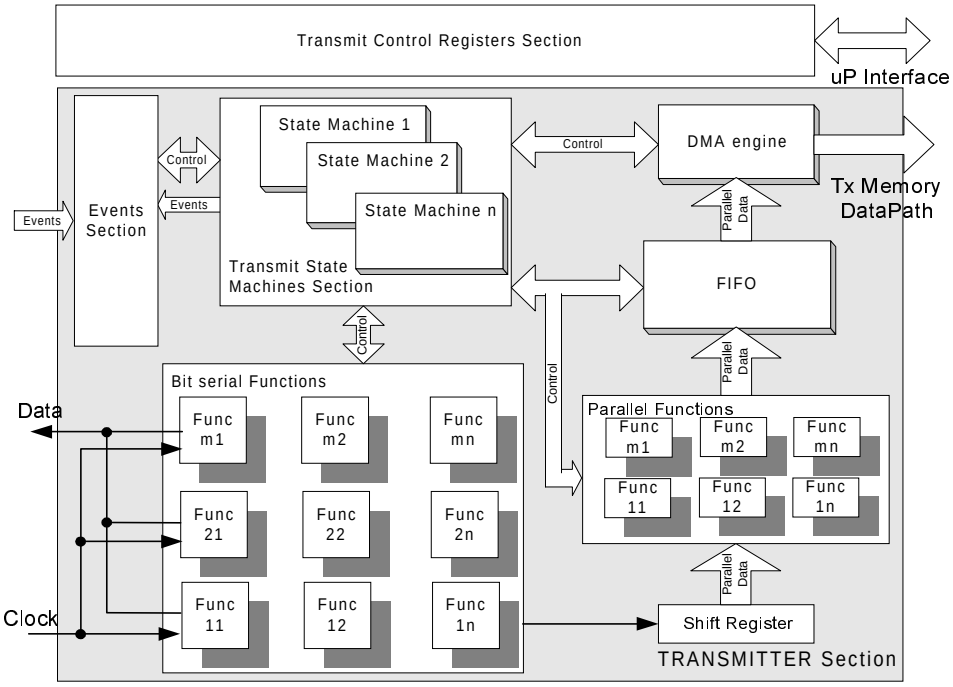

Figure 6. General Network Architecture-Transmitter

side the bit-serial functions block gets input from the network and gives output to the shift register while in the transmit side gets input from the shift register and outputs to the network. The parallel functions block contains an array of parallel functions connected with configurable interconnections as in the bit-serial functions block. The parallel functions block interfaces with the shift register and the FIFOs.

The events section monitors network events and informs the state machines section which controls and collects status from all the other blocks in the architecture. FIFOs are parameterized according to network buffering requirements and are connected to the DMA engine blocks or to the control registers section depending on the data path implementation.

The General Network Architecture described can be customized to MAC protocol requirements in order to produce the Customized Network Block which fits to a specific protocol needs. The customized network block together with a microprocessor and a memory interface can be easily integrated into an embedded MAC controller, for rapid development of communication systems (Figure 7).

As illustrated in Figure 7, the embedded MAC controller architecture consists of the customized network block that interfaces with the network physical device and the microprocessor. The separate datapath to the memory is optional and depends on the existence of a DMA block in the customized architecture. A memory controller interfaces the microprocessor and the customized network block with external memory devices offering a common memory space to both modules. The microprocessor implements all the control and management functions of the protocol and transacts with the customized network architecture by sending control and accepting status information.

The architecture can be completed with modules such as UARTs, ISA/PCMCIA interfaces to offer a complete system solution. Next section describes the application of the general network architecture to the implementation of a MAC processor for IEEE 802.11 networks.

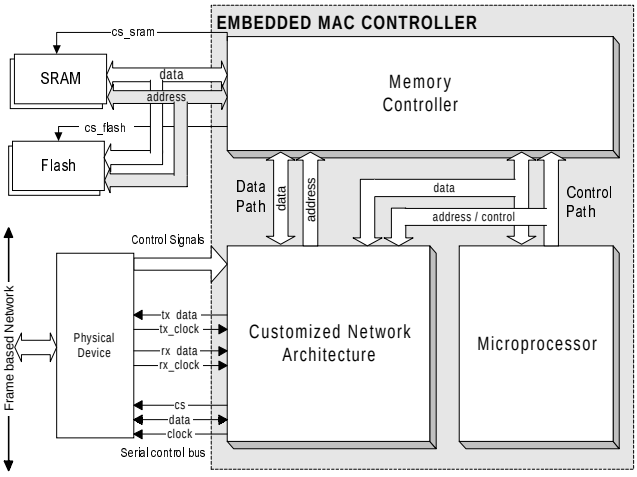

Figure 7. System Architecture

4. Application of GNA to the IEEE 802.11 MAC implementation

For the implementation of a MAC processor for the IEEE 802.11 protocol [4], the general network architecture should be customized as follows:

The bit serial functions required by the IEEE 802.11 are two CRC-32 engines, one for transmit direction and one for receive direction, which calculate the CRC on transmitted or received serial data. These bit operations do not alter the serial data that are fed to the shift register device.

The parallel functions in the IEEE 802.11 MAC are used to XOR the raw data with random numbers in both the transmit and receive sections for (optional) encryption/decryption, and to compare the packet address with predefined station address value (in the receive side) for recognizing a unicast, broadcast or multicast packet.

The events section recognizes events on Start of Frame, End of Frame (in the receiver), Start of Transmission, End of Transmission and Clear Channel Assessment (in the transmitter). Also the events processing block recognizes events on TSF register (which is a protocol defined register for synchronizing network events), DMA control register etc.

The control registers section contains registers for state machines, DMA programming, encryption/decryption programming, reading network status, synchronizing network events (TSF timer) etc. The FIFOs in the transmit and receive directions are 128-bytes long in order to offer appro-

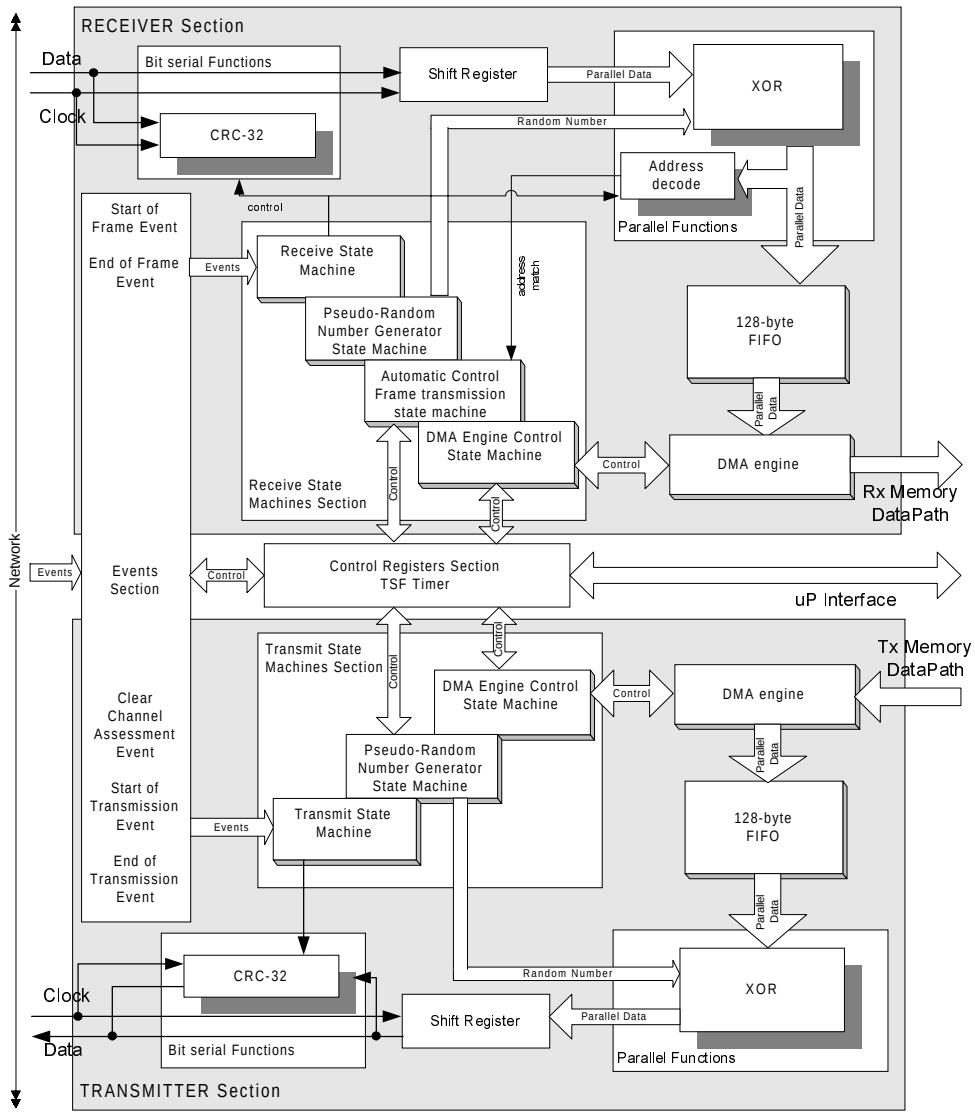

Figure 8. The Customized Network Architecture for IEEE 802.11 MAC implementation

priate buffering for speeds up to 11 Mbps.

In the receive direction there are four state machines. The receive state machine accepts the receive bytes and stores them in the FIFO. The random number generator state machine produces the random numbers that are XORed with received data in order to be decrypted (optional). The automatic control-frame transmission state machine automatically transmits control frames when it recognizes unicast address in a correct reception. The DMA engine control state machine transfers a block of data to the memory.

In the transmit direction there are three state machines. The transmit state machine accepts data from FIFO and transmits them over the network. The random number generator state machine produces the random numbers that are XORed with parrallel data in order to encrypt the transmitted data (optional). The DMA engine control state machine transfers a block of data from the memory to the transmit FIFO.

The customized network architecture described above combined with an ARM microprocessor core and supporting peripherals such as interrupt controller, PCMCIA interface, and timers, has been realized in an ASIC.

5. Conclusions

The methodology described in this paper allows the rapid development of communication systems by automatically generating the supporting hardware from a general parameterized architecture. This method has the advantages of a custom solution (efficiency, low power) while keeping the sense of flexibility and programmability of a microprocessor.

References

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

[2] Bluetooth Consortium, *Specification of the Bluetooth System*, version 1.0B, December 1999.

[3] ANSI/IEEE Std 802.3-1996: *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and physical layer specifications*.

[4] S. Koutroubinas, A. Maniatopoulos, M. Iliopoulos and T. Antonakopoulos, *V-NET: Design, Implementation and Demonstration of an Adapter Card for Wireless Local Area Networks*, Bordeaux , EMMSEC ‘98