Reprint

Reconfigurable Network Processors based on Field Programmable System Level Integrated Circuits

M. Iliopoulos and T. Antonakopoulos

The 10th International Conference on Field Programmable Logic and Applications – FPL2000

VILLACH, AUSTRIA, AUGUST 2000

Reconfigurable Network Processors Based on Field Programmable System Level Integrated Circuits

Marios Iliopoulos and Theodore Antonakopoulos

Department of Electrical Engineering and Computers Technology
University of Patras, 26500 Rio – Patras
Greece
antonako@ee.upatras.gr

Abstract. The increasing demand of networking applications has imposed a new category of electronic circuits that integrate powerful CPU processing, networking and system support functions in a single, low cost chip. These integrated circuits, called Network Processors, are optimized for tasks such as access protocol implementation, data queuing and forwarding, traffic shaping and Quality of Service (QoS) support. This paper presents the use of Field Programmable System Level Integrated Circuits that combine the flexibility of programmable cores and the high performance of dedicated hardware, to implement network processors used for medium access protocols.

1 Introduction

Network Processors should offer flexibility, programmability, performance and low cost while being able to shorten time-to-market cycles for new networking products. However, these requirements contradict since network processors that consist of programmable cores may offer flexibility and less time-to-market but usually they have poor performance and increased cost. On the other hand, network processors that contain dedicated hardware offer high performance and low cost, but are less flexible and have higher time-to-market cycles [1]. Another solution is to use a programmable core supported by dedicated hardware in order to increase the efficiency and performance of programmable cores. Although this solution is easily adaptable to newer versions of the supported protocol, it still suffers from decreased flexibility, since dedicated hardware restricts the usage of the chip to a specific application.

Latest trends in network processors design combine RISC processor cores with reconfigurable hardware in an attempt to solve the trade-off of high performance and flexibility. A new emerging technology developed towards this approach is called Field Programmable System Level Integrated Circuits (FPSLICs) [2] and combines an 8-bit RISC microcontroller (AVR), reconfigurable hardware using Field Programmable Gate Array (FPGA) cells and SRAM, thus it can offer a single chip solution for complete systems.

R.W. Hartenstein and H. Grünbacher (Eds.): FPL 2000, LNCS 1896, pp. 39–47, 2000.

This paper presents the use of FPSLIC architecture as a vehicle to implement reconfigurable network processors that are able to implement low complexity access protocols. This is because the 8-bit AVR processor is considered „weak“ for the demands of complex network protocols, but it can still be used for networking applications such as 10Mbps Ethernet MAC, Point-to-Point Protocol controllers, home control networking, etc. On the other hand, the same basic idea supported by a 32-bit powerful microprocessor (such as an ARM processor core) and more reconfigurable logic will enable the implementation of network processors capable of supporting higher data rates and more complex protocols such as the IEEE 802.11, Bluetooth and 100Mbps Ethernet.

This paper makes use of a parametric architecture called General Network Architecture (GNA) [3] that directly maps network functions into a set of customizable hardware blocks that are interconnected through flexible interfaces. Section 2 introduces the FPSLIC architecture, the general network architecture and how GNA is mapped to an FPSLIC device. Section 3 demonstrates the implementation of a 10Mbps Medium Access Controller using general network architecture and one FPSLIC device. Finally, section 4 describes the implementation of more powerful network devices using the FPSLIC architecture and distributed processing.

2 Introduction to FPSLIC and GNA Architectures

In order to understand the use of FPSLICs for implementing reconfigurable Medium Access processors, we will describe in brief the FPSLIC architecture (section 2.1), the General Network Architecture (section 2.2), and how the GNA architectural blocks map to the FPSLIC device resources.

2.1 The FPSLIC Architecture

As illustrated in Figure 1, the initial version of FPSLIC contains the AVR microprocessor core, which is an 8-bit RISC processor with single-clock instructions, approaching 1 MIPS/MHz and 32 general purpose registers. The AVR core is supported by peripherals such as flexible timer/counters, a Real-time Counter, UARTs, programmable Watchdog Timer with internal oscillator, a 2-wire serial port interface and programmable I/O ports. The FPSLIC also contains 36K bytes of SRAM for program execution and data storage.

The reconfigurable part of FPSLIC is an SRAM based FPGA module with configurable Dual Port RAM cells. The FPGA has user programmable I/Os for interfacing to external world and may support low to medium complex devices (10K to 40K gates).

The AVR and the FPGA module communicate using three different interfaces: a control interface, 16 interrupt lines and a dual port RAM. The control interface decodes AVR address lines and control signals for accessing 16 memory mapped registers implemented in the FPGA, thus it can be used for implementing custom peripherals and/or directly controlling FPGA functions. AVR’s interrupt controller can accept 16 programmable interrupt lines, which are produced by the FPGA.

Finally, the AVR and the FPGA can also communicate through a dual port RAM which can be read/written by both the AVR and the FPGA allowing shared memory implementations.

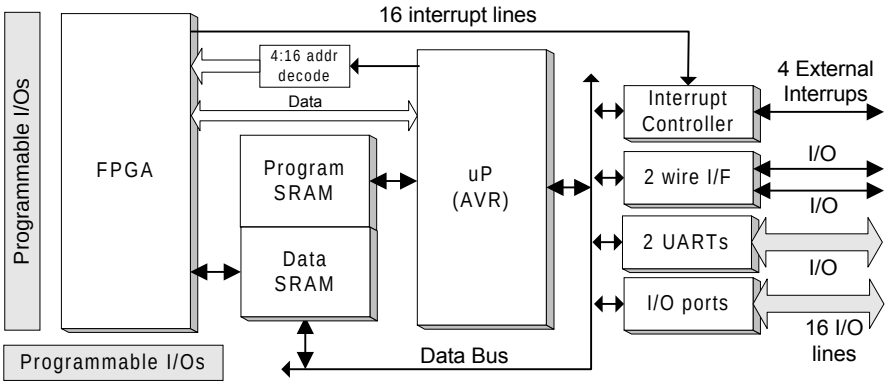

Fig. 1. The FPSLIC architecture

2.2 The General Network Architecture (GNA)

The FPSLIC resources are exploited by using the General Network Architecture (GNA) [3], which is a parametric architecture for realizing Medium Access Protocols. The General Network Architecture consists of customizable hardware blocks that are interconnected through flexible interfaces according to the dataflow illustrated in figure 2. The customizable hardware blocks perform *bit-serial functions* that process the serial bit-stream, *parallel functions* that process the parallel data, *event processing functions* that recognize the network events, and *control functions* that synchronize all the above blocks and consist of control registers, and state machines.

According to figure 2, the received serial data are passed through the bit-serial and parallel operations before they are stored into buffers and processed by the protocol's functions implemented in firmware. 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. The processor core configures/controls the GNA blocks and collects status information through a control interface.

There are two main blocks in the architecture, the Receiver section which contains all the receive related functions, and the Transmitter section that contains all the transmit related functions. 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.

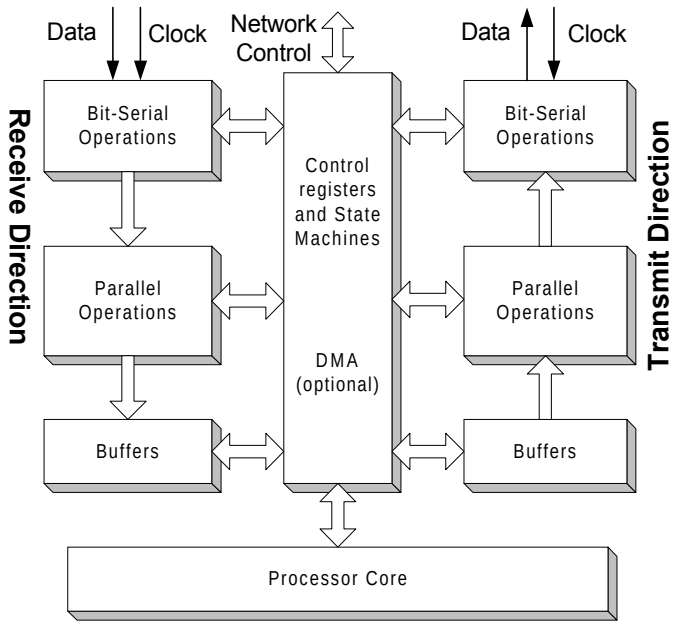

Fig. 2. The General Network Architecture

The data movement from/to the memory is accomplished through a dedicated path, either transparently without processor intervention by using a DMA engine, or by using direct processor read/writes without any DMA support.

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

Using the FPSLIC, all network related functions i.e. the bit-serial, parallel functions and the state machines contained in the GNA are implemented into the programmable logic, while the RISC processor implements all the control and management functions of the protocol. The FPSLIC's dual port SRAM, which is accessible by both the reconfigurable logic and the microprocessor can be used for temporary data storage while control and status information can be exchanged through the control and interrupt logic provided. Host transactions can be

accomplished either by using the integrated UARTs, or through a custom host interface implemented in the programmable logic. With the integration of the program SRAM, FPSLIC does not require external memory devices, except from a serial FLASH for downloading program data into its internal memory.

3 Application Example – 10 Mbps Ethernet MAC Controller

A 10Mbps Ethernet MAC controller is used to implement the access protocol as defined in [4]. In the receive direction the MAC controller searches for Start-of-Frame Delimiter (SFD) at the beginning of each packet, qualifies the frame length and the received address and verifies the CRC. Also, it observes any receive code violations or carrier events that may occur and keeps a status vector for each packet received. In the transmit direction the MAC controller adds to the supplied frame a seven-bytes preamble and one-byte SFD, performs padding on frames having fewer than 60 bytes and appends the CRC. Also, the MAC controller performs the carrier sense, collision detection and the back-off algorithm and reports the status of the transmission using a status vector for each packet.

The application of GNA and FPSLIC for the implementation of a 10Mbps MAC controller is illustrated in figure 3. The general network architecture is customized in this case for supporting a 10 Mbps Ethernet network interface card.

The network functions implemented in the reconfigurable part of the chip consist of a receiver section and a transmitter section. The receiver section consists of a CRC-32 check module (bit-serial function), preamble detect, SFD detect modules (parallel functions) and the receive state machine which controls all the receive blocks using information from the control registers and network events (carrier sense, collision detect module). It produces write enable and address signals for writing the received data in the common dual port RAM module. On the other hand, the transmitter section consists of a CRC-32 generation module and a transmit state machine which controls the data flow to the network. It receives information from the control registers and network events and produces address and control signals for reading data from the dual port RAM.

Queue management, statistics gathering, buffer descriptors, control frame generation/detection, back-off and other MAC functions are performed by state machines implemented in software and executed by the AVR microcontroller.

An ISA interface, implemented in the FPGA, offers the appropriate host interface for reading/writing data and passing control, configuration information to the MAC. The full system is completed by an external I^2C flash for AVR program downloading and power up FPGA configuration, plus an Ethernet 10Mbps physical device.

In a typical reception procedure using the FPSLIC-MAC the receive block recognizes the start of packet (preamble) and the start of data (SFD) and stores the incoming parallel data to a buffer in the Dual Port RAM while sending an interrupt to the AVR processor. The AVR processes the packet header and constructs a receive buffer descriptor in the dual port RAM memory. When reception is completed, AVR causes an interrupt to the host indicating that a valid packet is stored at the location indicated by the buffer descriptor. In the transmit direction, the host stores the data to be transmitted in the dual port RAM and constructs a transmit buffer descriptor. The AVR appends the preamble, the SFD and the padding if needed and initiates a

transmission. According to the status information received by the events processing block, it either marks the transmission as successful or retransmits the packet if a collision is detected performing the appropriate back-off algorithm. The transmission state machine appends the CRC and sends the data over the channel.

The implementation of the above 10Mbps MAC showed a problem in the FPSLIC architecture which was the limited addressing capability of the AVR processor to the FPGA part through the static microcontroller interface that uses only 2 address bits (directly address 4 registers and indirectly addresses 16 registers). A solution was to use the general purpose I/Os of the AVR processor to externally extend the address space for interfacing to the FPGA section.

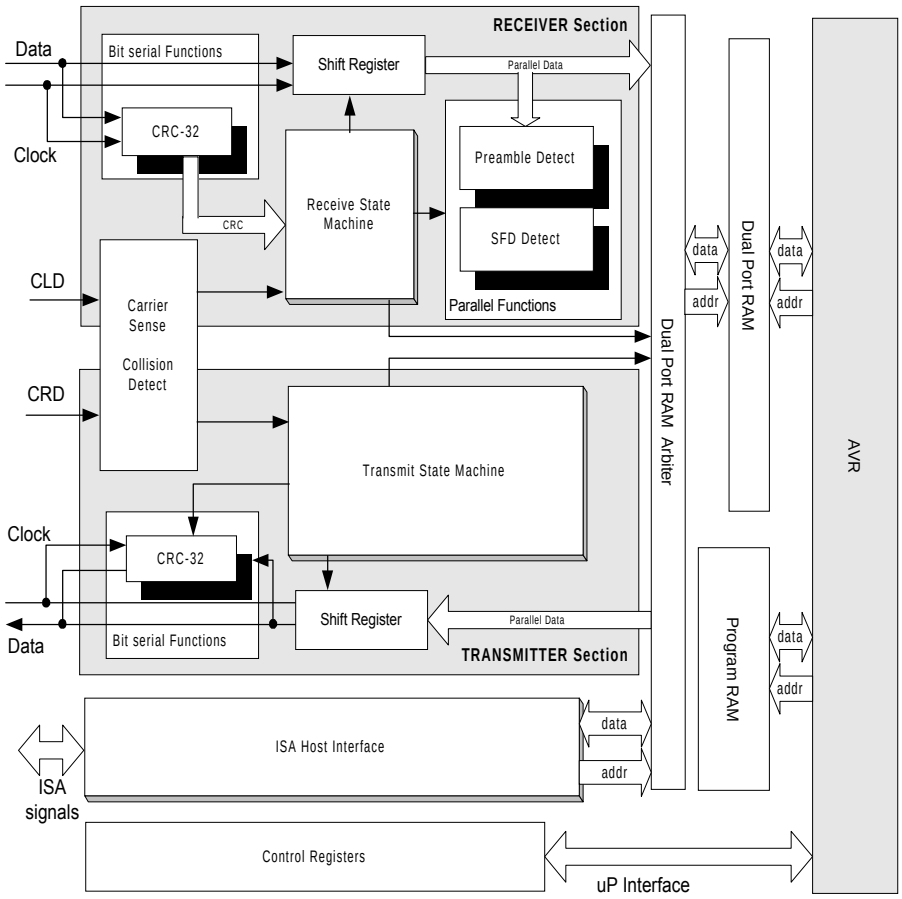

Fig. 3. Block diagram of an Ethernet MAC controller

4 Extending FPSLIC Capabilities

Due to the limited performance of the AVR 8-bit microcontroller, more demanding network architectures require powerful microcontrollers. One solution is to use more than one FPSLIC devices, implementing a distributed network processing solution. An architecture like the one illustrated in figure 4 is able to implement more complex network devices for processing at higher data rates and for supporting more demanding protocols.

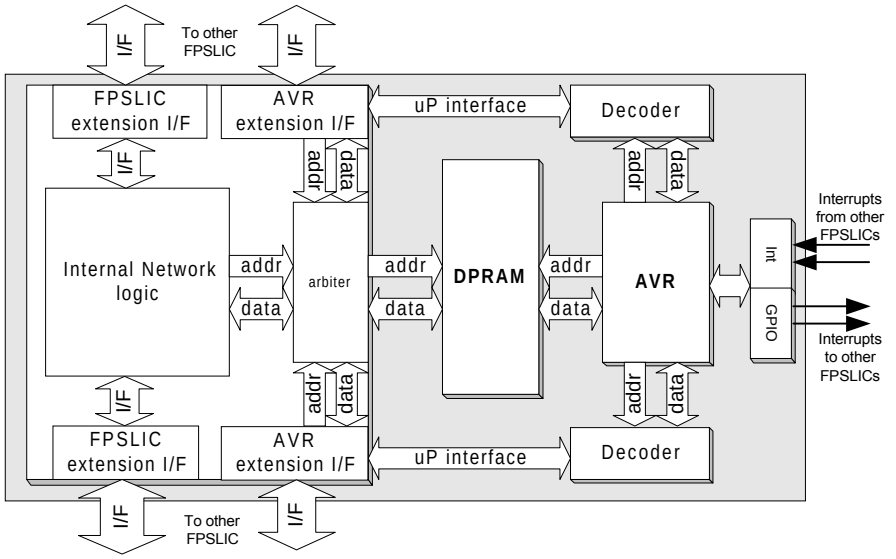

Fig. 4. FPSLIC configured for extension

The idea is based on connecting the configurable parts of two or more FPSLIC devices to produce a more powerful device with common memory space and to off-load protocol complexity by using more than one AVR processors. The communication between AVRs takes place through GPIOs, interrupts and a control path that is implemented in the reconfigurable logic illustrated as the AVR extension interface. Using the AVR extension interface, the AVR can also access the DPRAM of adjacent FPSLIC devices. The configurable part of the FPSLIC is extended through the FPSLIC extension interface.

An architecture containing two FPSLIC devices like the one shown in figure 5, can be used to implement access protocols requiring full duplex operation. Each AVR processor is attached to one direction (transmit or receive). The AVR processors execute code that controls the respective direction, while exchanging information through the AVR extension interfaces using either control registers or the dual port RAMs that can be accessed by both AVR cores (dashed line). A host interface can be implemented in the reconfigurable part of one of the FPSLIC devices while still having access to the dual port RAM of the other device through the FPSLIC extension interface.

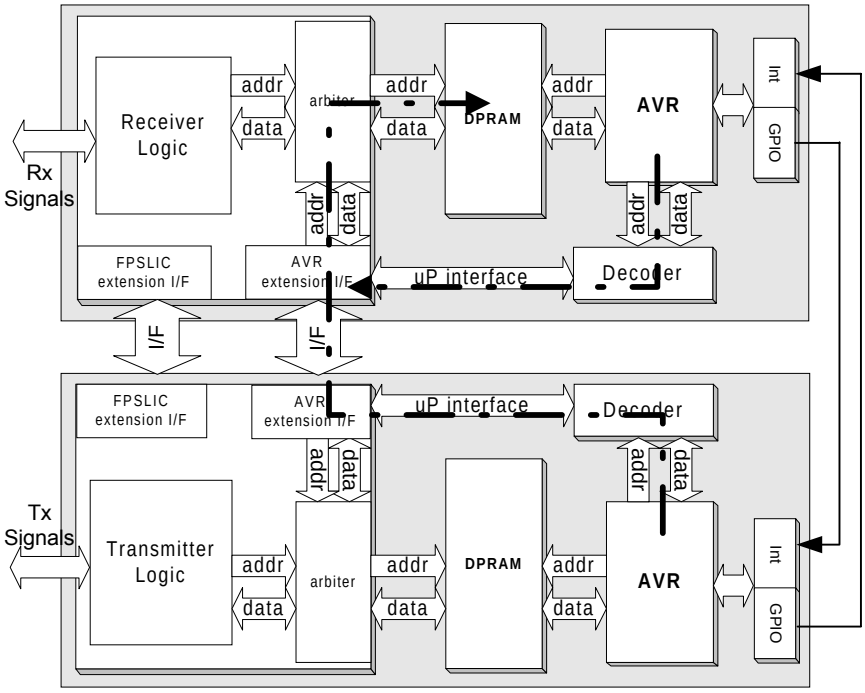

Fig. 5. Network processing example for the extended FPSLIC architecture

Two FPSLIC devices are connected together to implement the bridging between a modem connected to a telephone line and the Ethernet. The AVR processor of the FPSLIC device connected to the modem (through RS-232 interface), implements the PPP protocol and stores the data to the DPRAM of the FPSLIC connected to the Ethernet. The FPSLIC MAC processes the data according to the IEEE 802.3 protocol and transmits them over the Ethernet. In the other direction, the FPSLIC MAC processes the data from the Ethernet and passes them to the other FPSLIC device that performs the appropriate protocols in order to be sent over the telephone line.

5 Conclusions

The configurable nature of an FPSLIC device together with the general network architecture gives the network designer the flexibility to implement different access protocols based on the same platform, which consists of the microprocessor development tools and a HDL model of the General Network Architecture. In this paper we presented the use of FPSLIC architecture for implementing low complexity reconfigurable network processors and how this architecture can be extended to implement more powerful, distributed network processing tasks.

References

1. Nicholas, Cravotta, *Network processors: The Sky's the Limit*, EDN Magazine, November 1999, pages 108-119.
2. ATMEL, AT94 Series Field Programmable System Level Integrated Circuit, Advance Information.
3. Marios Iliopoulos, Theodore Antonakopoulos, *A Methodology of Implementing Medium Access Protocols Using a General Parameterized Architecture,* 11th IEEE International Workshop on Rapid System Prototyping, June 2000, France.
4. ANSI/IEEE Std 802.3-1996: *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and physical layer specifications*