Reprint

A PCI-bus based ASN.1 Accelerator

E. Ziouva, A. Avrasoglou, K. Agavanakis, T. Antonakopoulos, and V.Makios

The 6th International Conference on Advances in Communications and Control – COMCON'6

CORFU, GREECE, JUNE 23-27, 1997

The 6th International Conference on Advances in Communications and Control - COMCON 6
Corfu, Greece, 23-27 June 1997

A PCI-bus based ASN.1 Accelerator

E. Ziouva, A. Avrasoglou, K. Agavanakis, T. Antonakopoulos and V. Makios

Laboratory of Electromagnetics,
Department. of Electrical Engineers and Technology of Information
University of Patras, 26500 Rio - Patras
GREECE

Abstract: *The challenge to increase performance of processing of ASN.1/BER coded network protocols by hardware implementation is still valid. This work presents a PCI-bus based ASN.1 accelerator that can be used for speeding up the execution of various ASN.1 based protocols in application areas like electronic messaging systems and image processing devices. The accelerator is based on a high processing power architecture that incorporates an ARM core, an ASN.1 coprocessor and multiple local busses.*

1. Introduction

The advent of the open systems concept has led to the adoption of international standards (CCITT, ISO, ETSI, etc.) for data interchange between heterogeneous systems, both through transmission links and through storage media. Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER) are common ISO/CCITT standards concerning the encoding and decoding of data structures [1]. A hardware implementation of these standards is expected to substantially improve the performance characteristics of communications systems based on OSI (open system interconnection) standards. Implementing an ASN.1 coprocessor building block in the form of a "supercell" in the framework of the open microprocessor environment deployed by the Open Microprocessor Initiative (OMI), alleviates the processing burden and improves the responsiveness of all compute-intensive communications-oriented data processing applications. Additionally, users are able to create powerful communications-oriented single-chip systems, by combining ASN.1 coprocessor supercell with other peripheral supercells (of the cell library developed in the framework of the OMI program) implementing various types of network connectivity.

For demonstrating and evaluating the performance of the ASN.1 coprocessor (hardware-based approach) a subsystem had been developed in the form of a plug-in board for use in a personal computer or a workstation. The architecture of this subsystem is the subject of this work, which was performed in the framework of the ESPRIT-OMI ACCENT project 9169 - "ASN.1 Coprocessor Cell for the European Networking Technology" [2]. The aim of that project was to develop a supercell for a coprocessor implementing the encoding and decoding of data structures in accordance to the ASN.1 std. This supercell conforms to the interface standards defined by OMI for supercell interconnection and thus making possible to combine the coprocessor with host processor supercells and other peripherals for single-chip system configurations.

2. The Abstract Syntax Notation One (ASN.1) and the ACCENT Coprocessor

The function of ASN.1, together with the Basic Encoding Rules is to define the relationship between the meaning of abstract entities and their representation while being transmitted in a canonical, machine independent form [1]. ASN.1 defines a generalised set of standard notations for data types, each with a standard representation or encoding. For example, the standard types include strings, integers, and Booleans, and the standard representations define how each of these is in turn represented on the communications line as a series of octets. It is possible to use these types to build other more complex types corresponding to records or structures in programming languages. The standard representations correspond to the rules for converting between an internal representation of this format item and the external sequence of characters. ASN.1 uses a Backus-Naur notation to describe the various permissible forms. There is a set of built-in types, such as Boolean, Integer, Bitstring and OctetString. Then variations on these types can be defined, and in particular they can be grouped together into more complex combinations. Each value of each type is represented as a triple: an identifier, a length, and the contents. The Basic Encoding Rules are very detailed in specifying the order of the bits and bytes so that there should be no scope for different interpretation of the received protocol elements in the way we have seen in diverse computer systems.

Without using ASN.1, the normal approach is to consult some extra compiler information to determine how enumerated types map onto binary representations, and then to arrange the declarations accordingly. The explicit association of identifiers with numbers in ASN.1 avoids having to resort to such privileged information.

In order to implement these explicit associations and to accelerate the interpretation process, the ACCENT project developed an ASN.1 coprocessor for performing these functions in hardware.

The ACCENT coprocessor [3] uses two similar parts, the decoder and the encoder, together with certain resources that are shared between the two. The function and operation of each of the constituent blocks is summarised below.

- The protocol memory holds the binary representation of the ASN.1 protocols that the coprocessor must be capable of handling. This memory is a shared resource for the encoder and the decoder. It is 16 bits wide and its size is limited to 128 Kbytes.
- Separate micro-program memories are associated to the encoder and the decoder. These memories are accessed 1 byte at a time and their size is limited to 64 Kbytes.
- A block is used to execute all micro-instructions related to micro-program flow control and dispatches all other micro-instructions to the other blocks.
- Another block performs the arbitration between the encoder and the decoder for the allocation and release of shared resources.
- Logic operations and bit handling operations tailored to the processing of ASN.1 constructs are also included, and that block is shared between the encoder and the decoder.
- Comparison operands of various types 16 bits at a time, and an arithmetic unit, with a 32-bits adder and a 32-bits barrel shifter is also included.
- A block is used for managing the transactions on the coprocessor's external bus interface and for exchanging commands, responses, status information, etc. with an external host processor.

There are various busses that interconnect the coprocessor blocks:

- 16 bits wide input busses allow data exchange between various blocks,
- an 8 bits wide micro-program bus carries data fetched from the respective micro-program memories, while an 8 bits wide protocol bus is used to supply data fetched from the protocol memory.
- the coprocessor control bus propagates 12 bits of decoded micro-instructions, while a shared bus is used to distribute the information on the allocation of shared resources.

3. The ARM Processor

The ARM6 macrocell [4] is an implementation of the ARM microprocessor for use as a cell on application or customer specific integrated circuits (ASICS or CSICS). The ARM6 that has been used in the ASN.1 coprocessor, is a general purpose RISC processor which may be integrated onto an ASIC in conjunction with additional circuitry, e.g. logic, RAM, ROM and DSP cells. This 32-bit RISC processor macrocell supports Big and Little Endian operating modes and has 15 MIPS sustained performance at 25 MHz (25 MIPS peak). Its architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and related decode mechanism are greatly simplified compared with micro-programmed Complex Instruction Set Computers. This results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective chip.

The instruction set of ARM6 consists of ten basic instruction types. Two of these make use of the on-chip arithmetic logic unit (ALU), barrel shifter and multiplier to perform high-speed operations on the data in a bank of 27 registers, each 32 bits wide. Three instruction types control the transfer of data between main memory and the register bank, one optimised for flexibility of addressing, another for rapid context switching, and the third for indivisible semaphore operations. Two instructions control the flow and privilege level of execution, and the remaining three types are dedicated to the control of external coprocessors which allow the functionality of the instruction set to be extended off-chip in an open and uniform way.

Pipelining is employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

The memory interface has been designed to allow the performance potential to be realised without incurring high costs in the memory system. Speed critical control signals are pipelined to allow system control functions to be implemented in standard low-power logic, and these control signals facilitate the exploitation of the fast local access modes offered by industry standard dynamic RAMs. ARM6 is a fully static CMOS implementation of the ARM which allows the clock to be stopped in any part of the cycle with extremely low residual power consumption and no loss of state.

The data types the processor supports are Bytes (8 bits) and Words (32 bits), where words must be aligned to four byte boundaries. Instructions are exactly one word, and data operations are

only performed on work quantities. Load and store operations can transfer either bytes or words.

ARM6 supports six modes of operation:

- User mode: the normal program execution state
- FIQ mode: designed to support a data transfer of channel process
- IRQ mode: used for general purpose interrupt handling
- Supervisor mode: a protected ode for the operating system
- Abort mode: entered after a data or instruction prefetch abort
- Undefined mode: entered when an undefined instruction is executed

Mode changes may be made under software control or may be brought about by external interrupts or exception processing. Most application programs will execute in User mode. The other modes, known as privileged modes, will be entered to service interrupts or exceptions or to access protected resources.

ARM6 communicates with its memory system via two 32 bit data buses. It reads instructions and data from the input bus and writes data to memory on the output bus. In some applications, these buses can be directly connected to/from a bidirectional bus.

4. The PI Bus

The PI-Bus [5] is an on-chip bus to be used in modular, highly integrated microprocessors and micro-controllers (systems-on-chips). PI-Bus is designed for memory mapped data transfers between its bus agents. Bus agents are on-chip function blocks (modules), equipped with a PI-Bus interface and connected via PI-Bus signals. A PI-Bus agent acts as a PI-Bus master when it initiates data read or data write operations once bus ownership has been granted to the agent. A PI-Bus agent which is addressed at a PI-Bus operation acts as a PI-Bus slave when I performs the requested data read or write operation. Typical masters are processor modules, coprocessors, or DMA controllers. Typical slaves are on-chip memory and interfaces to the external world.

To operate, PI-Bus requires an additional bus controller which performs arbitration, address decoding, and time-out control functions. The bus controller may also be equipped with implementation dependent functionality, like slave access control features.

The PI-Bus protocol is oriented towards fast PI-Bus agent accesses as well as to a high transfer bandwidth. A low-overhead protocol guarantees short response times at PI-Bus accesses which are needed for time-critical applications. Multiple data transfers allow PI-Bus to operate a high bandwidth.

Macrocells with a PI-Bus interface can easily be integrated into a chip layout even if they are designed by different manufacturers. The potential bus agents require only a PI-Bus interface of low complexity. Since there is no concrete implementation specified, PI-Bus can be adapted to the individual requirements of the target chip design. E.g. the widths of the address and data bus may be varied.

The PI-Bus is designed with requirements of high-performance systems in mind. Main features of the PI-Bus are:

- Processor independent
- Demultiplexed operation
- Clock synchronous
- Peak transfer rate of 200 Mbytes/s (@ 50 MHz bus clock)
- Address and data bus scalable (up to 32 bits)
- 8-/16-/32-bit data accesses
- Broad range of transfer types from single to multiple data transfers
- Multimaster capability

5. The ASN.1 Accelerator

The ACCENT ASN.1 accelerator board has to use all the above mentioned components in order to achieve high processing power. Thus, the basic elements of a hardware subsystem for encoding and decoding of information, according to the ASN.1 standard, are:

- the ASN.1 coprocessor performing encoding and decoding of data tokens according to the specified ASN.1 grammar,
- a local memory to cache processed data tokens and structures to control operation of the ASN.1 coprocessor,
- a host processor controlling encoding and decoding processes as well as information interchange with the main system.

The PI bus connects functional blocks and offers a wide range of functions, with primary focus on the communication requirements of various peripherals. The subsystem, in turn, is connected to a main processor through a high-speed industry-standard bus, the PCI bus. The ASN.1 subsystem, depicted in Fig. 1, is integrated into the system environment.

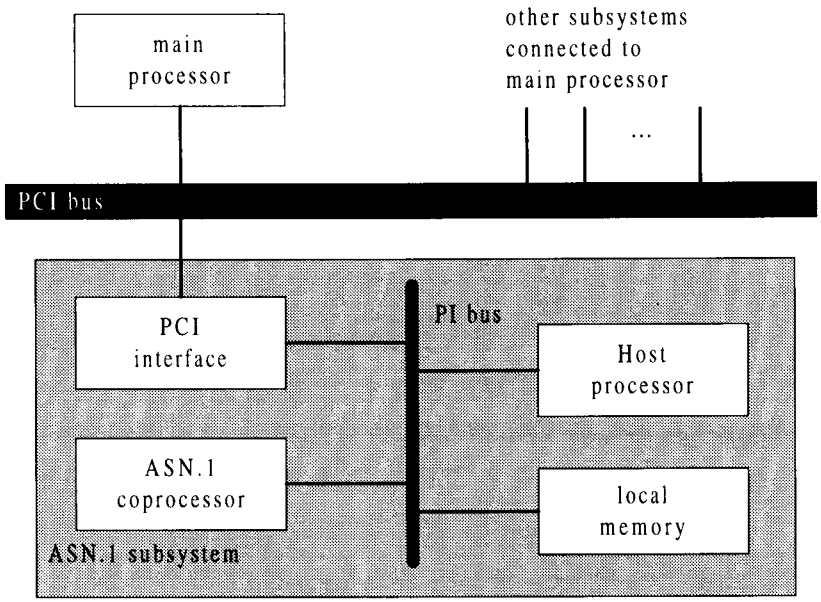

Fig. 1 The ASN.1 subsystem

For the subsystem board, different architectures have been considered, which reflect different approaches of interfacing board modules to the PI bus. The criteria for choosing the board architecture were: compliance to the PI bus standard and feasibility of implementation. Fig. 2 shows the system architecture in which only the ASN.1 coprocessor is equipped with an on-chip PI bus interface. All other modules are only connected to the relevant address and data lines of the PI bus and they are controlled in a module-specific way by the extended PI bus controller. The PI bus is designed with requirements of high-performance systems. Main features of the PI bus are processor independence, demultiplexed operation, address and data bus scalable up to 32 bits, 8/16/32-bit data accesses, board range of transfers types from single to multiple data transfers and multimaster capability. The subsystem board implements a 32-bit wide data bus and adopts big-endian byte ordering.

The ASN.1 coprocessor is either clocked by a 50 MHz local oscillator, or the 33 MHz PCI clock. In the first case, the coprocessor operates asynchronously to the PI bus clock, since its PI bus interface will synchronize between the internally used clock and the PI bus clock. In the second case, the coprocessor operates synchronous to the PI bus clock, since the PI bus clock is the PCI clock divided by 2 or 4. The grammar rules governing the encoding/decoding processes as well as the encoding/decoding micro-programs, are downloaded into local coprocessor memories: a memory area (32Kx8) for encoder micro-program, a memory area (32Kx8) for decoder micro-program and a memory area (128Kx16) for the protocol representation. The coprocessor can be a master or a slave of the PI bus.

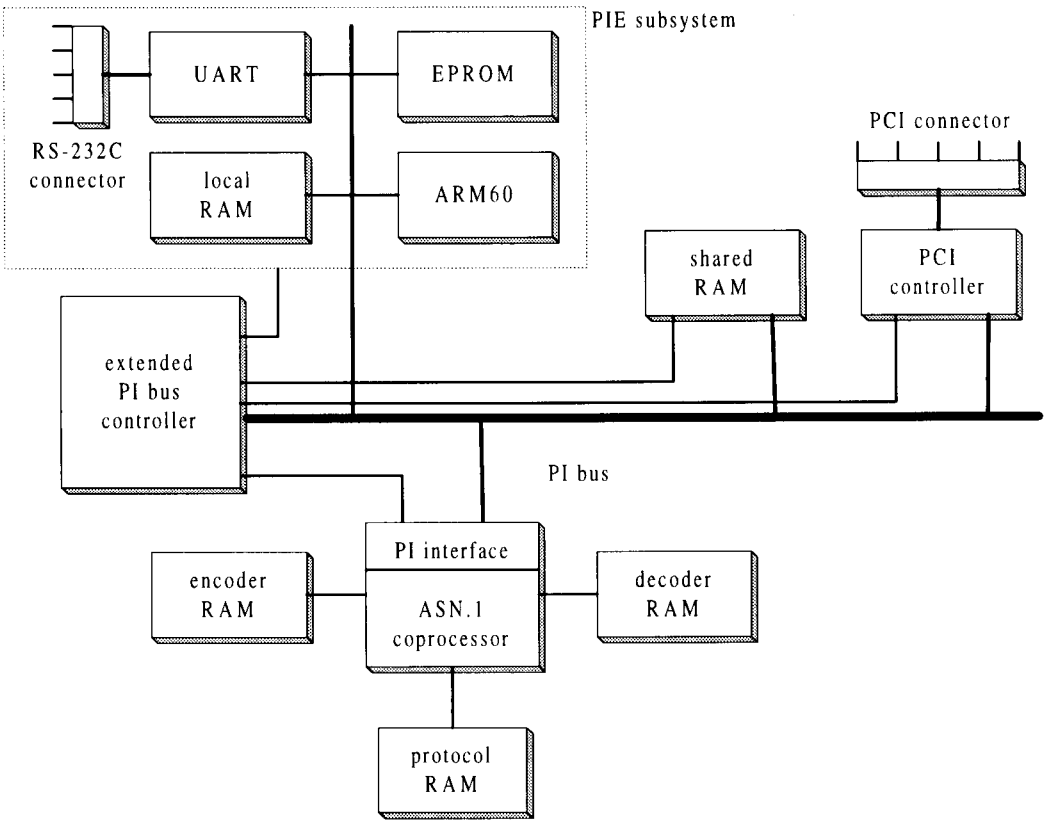

Fig. 2 The subsystem board architecture

As host processor the ARM6 processor is used, which is designed specifically for embedded applications. The ARM6 processor has exclusive access to: a 128Kx32 RAM area for data and program storage, a 128Kx8 EPROM for bootstrap and self-test firmware and a UART for serial

communications. The ARM PIE subsystem is also a convenient means of evaluating the ASN.1 subsystem board. The ARM6 processor is clocked by the PI bus clock and can be a master of the PI bus.

The shared memory area (SRAM) is used to exchange data between the ARM6 host processor and the ASN.1 coprocessor, and the subsystem and the main system via the PCI controller, and is made up of a single module with size 256Kx32. The subsystem board allows to mount two of these modules. The shared RAM can be a slave of the PI bus.

The main system and the subsystem can exchange data, control and status information over a powerful and flexible chip, the AMCC S5933 PCI controller. With a 32-bit PCI bus the S5933 can attain the peak transfer capabilities of 132 MByte/sec. The PCI controller can be a slave of the PI bus.

Finally, the extended PI controller does not only implement PI bus controller functionality, it also implements PI bus interfaces to various board modules; it controls and monitors the PCI controller, the shared RAM and the ARM6 host processor. This extended PI bus controller is implemented in an XC4010 FPGA and is described with more details below.

This architecture allows to use devices that are not compliant to the PI bus standard. So the criterion for compliance to the PI bus standard is satisfied by implementing the PI bus interfaces for each module in the extended PI bus controller. Furthermore, board space is saved since PI bus interfaces are integrated with other functional blocks into the extended PI bus controller, meeting the criterion for feasibility of the implementation.

PI Bus Controller implementation: The extended PI bus controller is implemented by an XC4010 FPGA. The block diagram of the extended PI bus controller is outlined in Fig. 3. This controller can function as a PI bus controller, a PI bus master and a PI bus slave. As a PI bus controller, the FPGA performs the following tasks: arbitration for bus ownership, address decoding/slave module selection and a time-out control. Bus control has to arbitrate which master is granted the requested bus ownership in accordance to the priority assignments for the PI bus master modules. If no master requests the bus, grant is given by default to one master. The address decoding mechanism determines the target slave of a bus operation by decoding the upper bits of the address issued by the granted master. The time-out mechanism is intended for bus operations which are not completed by the selected slave.

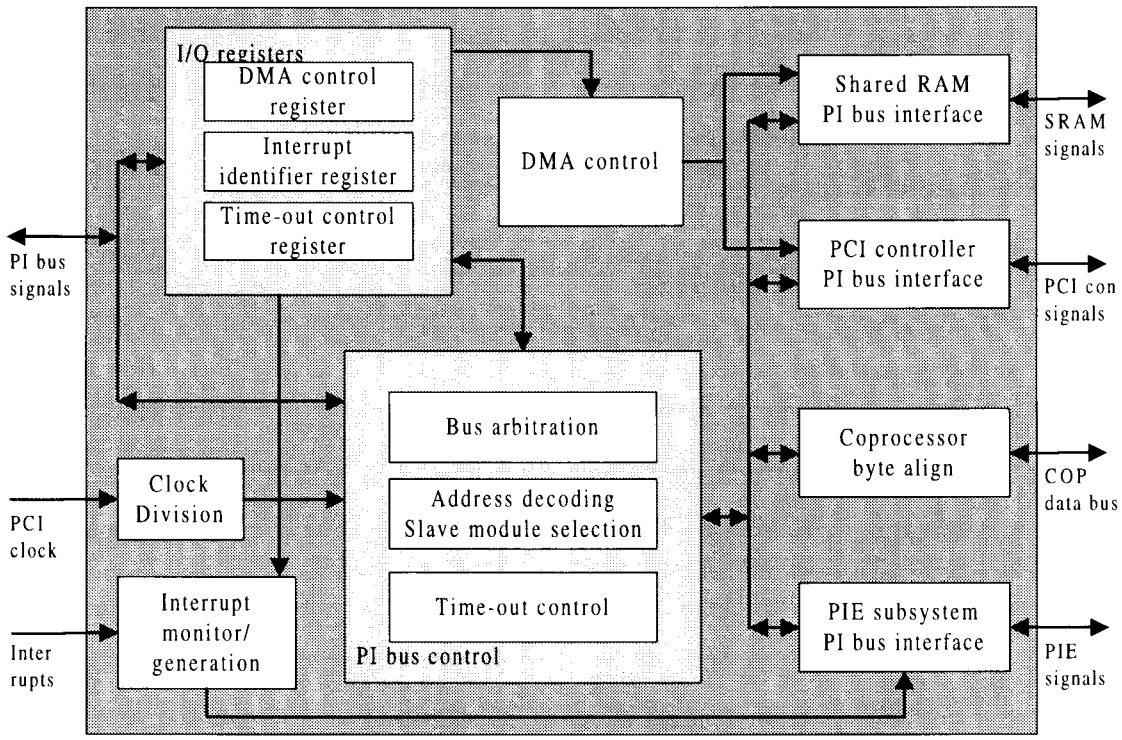

Fig. 3 Extended PI bus controller block diagram

As a PI bus master, the FPGA implements DMA transfers between the PCI controller and the shared RAM. In this case, the FPGA is not a true PI bus master since it does not read or write data itself, but it controls the PCI controller and the shared RAM to make these modules exchange data. As a PCI bus slave, the FPGA allows access to its I/O registers. These registers are a DMA control register, an interrupt identifier register and a time-out control register. The ARM6 processor may trigger the extended PI bus controller to operate as a DMA controller by writing the DMA control register. The FPGA may generate an interrupt to the ARM6 processor and then the processor has to read the interrupt identifier register to find out which device was the interrupt source (the PCI controller, the ASN. 1 coprocessor, the extended PI bus controller). The time-out control register may be written by a PI bus master module to configure the number of PI bus clock cycles that triggers a time-out event. Furthermore, the extended PI bus controller implements the PI bus slave interface of the PCI controller and the shared RAM and the PI bus master interface of the ARM PIE subsystem, since these devices are not compliant to the PI bus standard. It also implements an alignment on the ASN.1

coprocessor data bus, since the coprocessor implements right alignment on its data bus and the extended PI bus controller must take care to locate the bytes on their natural byte lanes. Finally, the extended PI bus controller divides the PCI clock by 2 or 4 to generate the PI bus clock.

The presented ASN.1 accelerator has been fully tested and evaluated and it is used by various European companies for building applications in the areas of electronic messaging and image coding.

6. Conclusions

An ASN.1 acceleration subsystem has been implemented by incorporating the ASN.1 coprocessor, a host processor, common memory and an interface to PCI bus. This subsystem is usable for various types of applications, differentiated by the software downloaded to the subsystem. The ASN.1 accelerator has been used in a message handling system for speeding up the document processing, message handling and system management functions. It has also been used for encoding/decoding image data according to the IPI-IIF ASN.1 grammar.

Acknowledgments

This work was partially supported by the European Union (EU) in the framework of the ESPRIT/OMI, ACCENT project.

References

[1] D. Russell: "The Principles of Computer Networking", Cambridge University Press, 1989.

[2] ESPRIT-OMI "ACCENT" Project 9169: "Technical Annex", 1994.

[3] A. Mathieu and H. Smit: "ASN.1 coprocessor hardware design specification", WP3/B3I/0026, Paris 1994.

[4] GEC Plessey: "The ARM6 Data Sheet", 1995

[5] Siemens AG: "OMI 324: PI Bus" Rev. 0.3d, Draft, Munich 1994