An FPGA-based Implementation and Simulation of the AAL Type 2 Receiver

Matti Tommiska, Mika Loukola, and Tero Koskivirta

Abstract: This paper describes a hardware implementation of an ATM Adaptation Layer (AAL) type 2 receiver. The simulation performance of the developed prototype is measured with real AAL type 2 encoded material and compared to two software-based AAL type 2 receiver implementations. The advantages and disadvantages of hardware-based and software-based simulation approaches in product development and prototyping are discussed. It is concluded that simulating a communication block directly in hardware both speeds up the simulations and decreases the development time.

Index Terms: AAL Type 2, Decoder, FPGA.

I. INTRODUCTION

The AAL type 2 has aroused interest among cellular network suppliers as a means for transporting data and voice between the base station and the mobile switching center (MSC) [1]. The bit rate from a mobile source is typically low and that is why a small size AAL type 2 Common Part Sublayer (CPS) packet can increase the overall performance as it provides bandwidth-efficient transmission of low-rate, short, and variable length packets in delay sensitive applications [2],[3].

The role of the AAL type 2 in mobile infrastructure networks is discussed in [2]. It will be applied first in W-CDMA diversity connections between the base station and the diversity combining point. Next it will be used also in MSC-MSC connections, thus bypassing transcoders.

Compared with other multiplexing methods, e.g., [4], the AAL type 2 is less bit efficient, but more tolerant to cell losses.

II. AAL TYPE 2 SPECIFICATION

As is the case with other ATM Adaptation Layers (AALs), the AAL type 2 is also subdivided into the Common Part Sublayer (CPS) and the Service Specific Convergence Sublayer (SSCS).

The AAL type 2 transfers AAL Service Data Units (AAL-SDUs) from one AAL Service Access Point (AAL-SAP) to another. The user can select among different AAL-SAPs (associated with SSCSs) that offer different Quality of Service (QoS). The CPS transmitter process receives CPS-SDUs from SSCS transmitter processes, and multiplexes and packs CPS-Packets (see Fig. 1). At the receiving end the CPS-Packets are subsequently unpacked, demultiplexed and passed to one of the SSCS receiver processes. The introduction of several QoS levels within a single ATM connection brings along the required switching capability of AAL-SDUs. This means that the intermediate ATM switches need to decode the AAL-SDUs from the incoming ATM cells and switch them according to the QoS requirements as additional feeds are inserted to the channel. The scheduling algorithms for QoS levels are not a part of the AAL type specification [2].

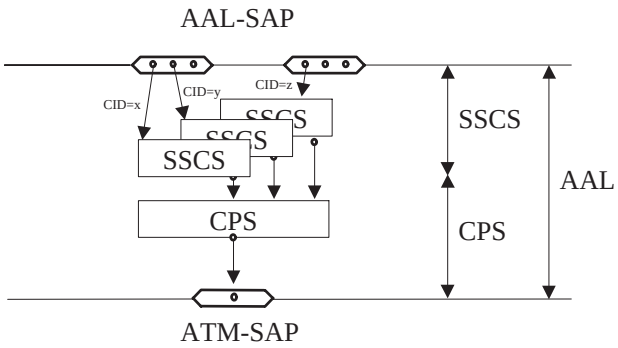

Fig. 1. Functional model of AAL Type 2.

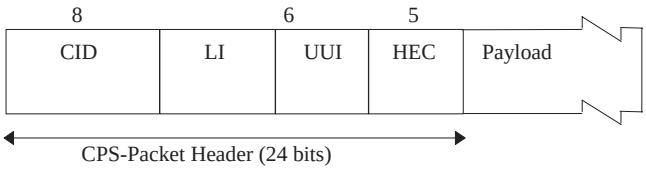

Fig. 2. Format of AAL Type 2 CPS-Packet.

The format of the AAL type 2 CPS-SDU, i.e., the CPS-packet is shown in Fig. 2.

The Channel Identifier (CID) identifies the AAL type 2 CPS user. As the channels are bi-directional the CID values are used for both directions. The Length Indicator (LI) field carries a value that is one less than the number of octets in the CPS-Packet payload. The maximum length is set by signaling or management procedures with the default value being 45 octets. The CPS users, SSCS entities, or Layer Management can transfer codepoint messages through the User-to-User Indication (UUI) field. The last CPS-Packet header field is the Header Error Control (HEC) field [2].

The CPS-PDU is shown in Fig. 3. The Start Field (STF) is subdivided into Offset Field (OSF), Sequence Number (SN), and Parity (P). OSF indicates the number of octets between the end of the STF and the start of the first CPS-Packet. SN bit is the modulo 2 number of the CPS-PDU stream. The parity of the STF field must be odd. The CPS-packet consists of the CPS packet header (CPS-PH) and payload.

Manuscript received August 31, 1998; approved for publication by Peter T-S. Yum, Division III Editor, November 17, 1998.

M. Tommiska and M. Loukola are with the Laboratory of Signal Processing and Computer Technology, Helsinki University of Technology (HUT), Finland, e-mail: Matti.Tommiska@hut.fi and Mika.Loukola@hut.fi.

T. Koskivirta is with the Nokia Research Center, Finland, e-mail: Tero.Koskivirta@research.nokia.com.

1229-2370/99/$10.00

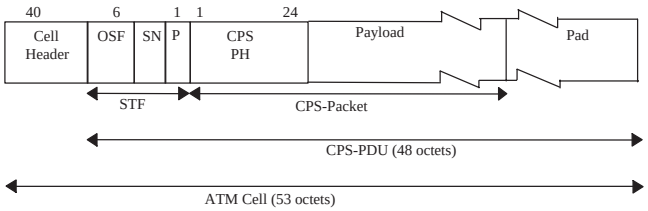

Fig. 3. Format of CPS-PDU.

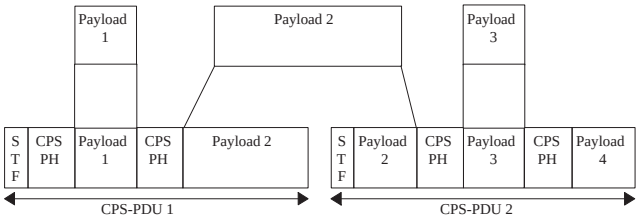

Fig. 4. Decoding of AAL Type 2 CPS-PDUs.

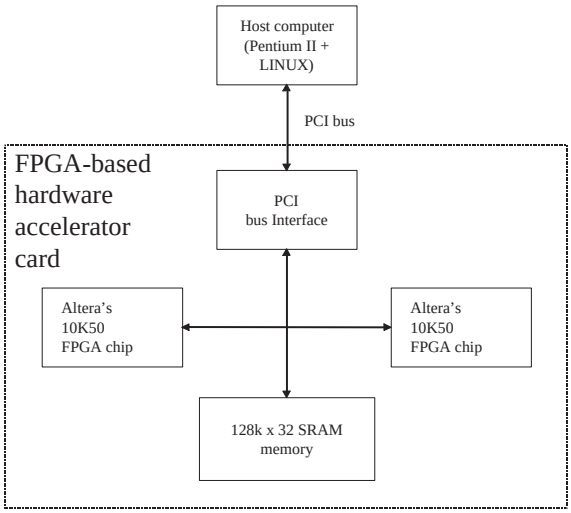

Fig. 5. Block diagram of the FPGA development environment.

Fig. 4 illustrates how a CPS-Packet may overlap one or two ATM cell boundaries. A CPS-Packet may be partitioned anywhere. If some cells have been dropped the receiver can resume decoding as the STF field points to the starting point of the next CPS-Packet. In the second CPS-PDU the STF field points to the third CPS-Packet which is the starting point of the first new CPS-packet in the second cell.

The source switch must carefully decide when to wait for additional AAL-SDUs and when to pad the remaining part of the CPS-PDU and transmit the cell. A compromise has to be made, since padding reduces the bandwidth-efficiency but at the same time decreases the delay characteristics of the channel. This problem is addressed in [5]. In case the CPS-Packet does not fill the CPS-PDU and the cell must be transmitted immediately the rest of the CPS-PDU is filled with zero-padded octets as illustrated in Fig. 3. A timer is associated with the transmitter process. When the timer expires the CPS-PDU is transmitted whether or not it has been filled up. The timer is reset when a CPS-Packet overlaps to the next CPS-PDU or when the a new CPS-PDU is initialized.

Upon receipt of a CPS-PDU the parity (P) indicated in STF is verified as well as the sequence number (SN). After that the HEC is verified. If no errors are detected then the receiver resumes to decode a possible overlapping packet followed by a new CPS-packet from the OSF point.

III. THE FPGA DEVELOPMENT ENVIRONMENT

The prototyping environment consists of a personal computer (PC) with a 233 MHz Pentium II processor and Red Hat 4.2 Biltmore LINUX as the operating system. The host computer is equipped with a special hardware acceleration card containing three Field Programmable Gate Array (FPGA) chips. Two of the FPGA chips are Altera's 10K50 devices, each of which have approximately 50000 gates. They are well suited for intensive bit-level calculations due to the fine-grained granularity of Altera's 10K family [6]. The third FPGA chip serves as a Peripheral Component Interconnect (PCI) bus interface.

In addition to the internal embedded memory of the FPGA chips, the hardware acceleration card has 128 kilowords of additional fast Static Random Access Memory (SRAM), which can be written to and read from both by the hardware acceleration card and the host computer. The most natural uses for the SRAM block are the exchange of information between the host computer and the hardware acceleration card and the storage of temporary results during the FPGA computation. The development environment encompasses also a custom-written loading and debugging software package, which enables the (re)programming of the FPGA chips and inspection of the shared 128 kilowords large memory. The architecture of e development environment is illustrated in Fig. 5.

IV. FPGA IMPLEMENTATION

The implementation of the first working version of an FPGA-based AAL type 2 receiver was a relatively straightforward procedure, since the underlying receiver state machine has been thoroughly described in [2]. However, the tuning and optimization of the sequential flow chart presented in [2] for a parallel hardware implementation was the most intensive and demanding part of the FPGA design process.

The design flow is described in Fig. 6. The first stage was to write a functional VHSIC Hardware Description Language (VHDL) [7] model of the AAL type 2 receiver and simulate it with Model Technology's ModelSim VHDL simulator. An alternative would have been to write a synthesizable VHDL description with Altera's Max+Plus II design software right away.

This approach might appear faster at first sight, but experience has taught that the time invested in first writing a functional VHDL model greatly helps in both discovering design flaws at the earliest possible stage and in understanding the overall algorithm.

The translation of a functional VHDL model into synthesizable VHDL is not an automated one, since the subset of VHDL that is synthesizable, i.e., can be compiled into a gate-level description, varies greatly among manufacturers of logic synthesis software. However, the translation process was not a major hurdle in this particular case, since the AAL type 2 receiver process

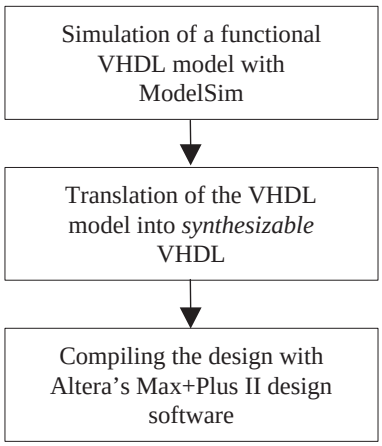

Fig. 6. Design flow of the FPGA-based AAL Type 2 receiver.

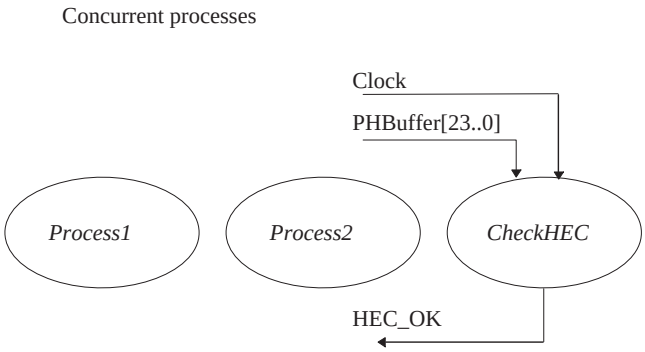

Fig. 7. AAL Type 2 CPS-PH (Common Part Sublayer Packet Header).

consists mostly of atomic bit-level operations.

The main reason for implementing the AAL type 2 receiver in reprogrammable hardware was to speed up simulations of the performance of a AAL type 2 receiver process, which in turn decreases the development cycle. This was accomplished by taking advantage of the parallelizability of the AAL type 2 receiver state machine. Independent operations can be performed simultaneously in "malleable" FPGA-based hardware, whereas a sequential microprocessor-based implementation is bound by the fixed instruction set and register-level implementation of the processor.

As a specific example of the advantages offered by an FPGA-based implementation over a software-based one, the Header Error Check (HEC) of the CPS-Packet Header (CPS-PH) is described.

The 5 bits of the HEC field (See Fig. 7) are the remainder of the division (modulo 2), by the generator polynomial x^5+x^2+1, of the product x^5 and the contents of the first 19 bits of the CPS-PH [2].

The AAL type 2 receiver has to check for the correctness of the HEC field in a received CPS-PH. To do this in software requires both unusual word-widths (19 and 5, respectively) and tedious sequential bitwise XOR operations. On the contrary, the VHDL description checks the HEC field in a unique process called CheckHEC, which is executed concurrently with other processes in the hardware (see Fig. 8).

The input to the CheckHEC process is the 24-bit wide PH-

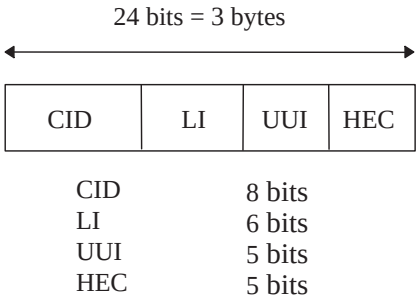

Fig. 8. Example of concurrency in hardware: Checking the HEC Field in CPS-PH.

Buffer, and the output HEC_OK is always available within the same clock cycle, since each bit of the 5-bit wide syndrome can be calculated by a parallel combinatorial XOR logic circuit [8]. The speedup achieved by this is of considerable amount, and it is exactly in this kind of bit-level operations where the advantages of programmable hardware are evident.

V. PERFORMANCE MEASUREMENTS

To compare the relative performance of the FPGA-based AAL type 2 receiver, two software versions of the AAL type 2 receiver were also written. The first version was written in the C programming language, which was complied and run under LINUX operating system. The other version was written as a MATLAB m-file and its relative performance was measured by running it under Windows95 operating system.

The results obtained from these comparisons can be used as a guideline for evaluating the quantitative speed advantage of hardware-based simulation over the one based on software. Both software-based versions of the AAL type 2 receiver read AAL type 2 encoded data from a file and write the output to files named channelxxx.txt, where the numbers xxx identify the particular channel. Since the Channel Identification (CID) field is 8 bits long, there is a maximum of 256 different channels available, but CID value 0 is not used and CID value 1 is reserved for Layer Management peer-to-peer procedures. CID values from 2 to 7 are reserved [2].

The C programming language version of the AAL type 2 receiver can handle simulated transmission rates up to 790 kbits per second when run on 200 MHz Pentium II PC. The actual transmission rate varies considerably between different 48 bytes long ATM packets, since the contents of each ATM packet differ substantially from each other. For example, an ATM packet may contain only the Start Field (STF) and 47 bytes of payload associated with a single channel. In this case, the decoding of the whole ATM packet is both easy and fast. If, on the other hand, an ATM packet contained numerous 3 bytes long CPS-packet headers, a considerable amount of time would be needed for the correct processing of each CPS-packet header. In this case, the simulated transmission rate is quite low when compared to "simpler" ATM packets.

For the reasons mentioned above, the simulated transmission rates must not be regarded as absolute values, but rather as an average obtained by simulating a set of widely varying pack-

Table 1. Performance comparison for different AAL Type 2 receiver implementations.

AAL type 2 receiver performance

	Hardware-based	Software-based	
	FPGA	C language	MATLAB
Simulated Transmission rate	55 Mbps	790 kbps	1 kbps
Notes:	Fclk = 23.7 MHz	Pentium II @200 MHz	Pentium @100MHz

Mbps = Megabits per second
Kbps = Kilobits per second

ets. The MATLAB version of the AAL type 2 receiver achieved a simulated transmission rate of 1 kbits per second when run on a 100 MHz Pentium PC. To make the simulated transmission rate comparable to that obtained by running the compiled C programming language version, the simulated transmission rate of the MATLAB version can be normalized to 2 kbits per second (200 MHz vs. 100 MHz processor clock frequency), if the somewhat simplistic assumption about the irrelevance of the operating system in use is made. This indicates that compiled (i.e., C source) code runs approximately 400 times faster than interpreted (i.e., MATLAB) code in this particular case.

As could be expected, The FPGA-based AAL type 2 receiver achieved the fastest transmission rates. According to the timing analyzer of Altera's Max+Plus II design software, the maximum clock frequency of the AAL type 2 receiver was 23.7 MHz, which may appear low when compared to the clock frequencies of leading microprocessors. However, due to the inherent parallelism of hardware and the suitability FPGAs for bit-level operations, the simulated transmission rate for the FPGA-based AAL type 2 receiver was 55 Mbits per second for the same set of AAL type 2 encoded data that was used with the two software-based versions.

The simulated transmission rates are summarized in Table 1. As can be seen from the table, FPGA-based simulation was orders of magnitude faster than the two software-based versions. The difference between the two software-based versions is also quite striking, but it is not by any means unexpected, since MATLAB m-files are interpreted, whereas C source files are compiled into machine code.

VI. COMPARISON OF HARDWARE AND SOFTWARE-BASED SIMULATION

When compared to software-based simulation, the main advantage of hardware-based simulation is speed. Hardware-based simulation allows the simulation process to take advantage of the parallel execution of instructions. For example, multiple independent assignment statements in a sequential software program can be performed in programmable hardware within the same clock cycle.

Other advantages of programmable hardware are the ability to perform bit-level operations on "unusual", i.e., not powers of two, word lengths and the possibility to allocate only a certain number of bits to represent internal variables. An example of atomic bit-level operations is the header error check process in an FPGA-based AAL type 2 receiver, where the 5-bit long syndrome is calculated within one clock cycle by a combination of bit-level XOR operations. A good example of the possibility to allocate only a certain number of bits to internal variables is the AAL type 2 receiver, where all internal variables can be represented by a maximum of nine bits, which means a big saving in hardware resources.

The design and construction of hardware-based simulation platforms has traditionally been a tedious undertaking, and this approach has been rarely employed in practice. However, this state of affairs has been somewhat changed with the advent of reprogrammable logic devices. This has enabled the construction of generic FPGA boards, which can be quickly customized to the particular simulation task at hand by reprogramming the FPGA chips. In other words, there is no need to design and construct a custom hardwired electronic printed circuit board (PCB) every time when hardware-based simulation is desired.

The benefits of hardware-based simulation are also evident, if the simulated algorithm is going to be implemented in a Application Specific Integrated Circuit (ASIC) for mass production. This is especially the case, if the hardware-based simulation process has been conducted with FPGAs and a hardware description language (HDL), for example, VHDL or Verilog, has been used in the design.

There are two great barriers for the use of hardware-based simulation, namely the undeveloped state of design tools and the steep learning curve associated with hardware design. Unfortunately, both of these obstacles will probably be preserved for some time, since there does not seem to be enough market pull nor technology push to make hardware-based simulation more accessible to people, whose background is in software development.

Software-based simulation has naturally its own advantages, too. Most people are used to at least one programming language, and preliminary results are usually obtained many times faster than when resorting to hardware-based simulation. If the simulated task is not large or is not a part of a larger entity, there usually is no need to speed up simulations by resorting to hardware acceleration.

A comprehensive simulation of all the different parameters in a modern communications system and the interplay between the individual (e.g. transmitter, receiver, coder) blocks is extremely time consuming in software, but a parametrizable FPGA-based hardware accelerator may provide enormous speedup ratios. The addition of truly analog noise sources for example, with a noise diode is an added bonus in hardware-based prototyping, since there is no need to resort to pseudorandom bit generators, which are necessary in software-based simulations.

The advantages and disadvantages of hardware-based and software-based approaches to simulation are summarized in Table 2.

VII. CONCLUSIONS

The significance of AAL type 2 is in its ability to achieve high bandwidth efficiency and low packetization delay at the same time [9]. The essential characteristics of AAL type 2 have been standardized, and its potential application areas include packet telephony applications over backbone ATM networks. It is desirable to keep the packet sizes smaller than the default payload

Table 2. Comparing hardware and software-based approaches to simulation.

Property	Hardware-based	Software-based
Simulation speed	Fast	Slower, especially with interpreted languages
Parallelism	Inherent	Bound by the sequential nature of the microprocessor
Bit-level operations	Easy	Tedious
Design cycle	Longer, but getting shorter	Short
Design tools	Still immature, long learning curve	Mature

size of 48 bytes.

An FPGA-based AAL type 2 receiver was designed, and its simulation performance was found to be orders of magnitude faster than that of two software-based versions. An FPGA-based development environment was found to speed up both the simulation and the prototyping by a substantial factor. The benefits of hardware-based simulation, especially on reprogrammable hardware, are its speed advantage and convertibility into production versions of the simulated design. The spreading use of HDLs in FPGA and ASIC design has helped to bridge the gap between hardware and software, but nevertheless, the ability to "think hardware, write software" is still a quite rare phenomenon.

REFERENCES

[1] M. Han and A. A. Nilsson, "Simulation study of AAL type 2," in *Proc. 1st International Conference on ATM (ICATM'98)*, Colmar, France, June 1998.

[2] P. Schicker, "ITU-T recommendation I.363.2 B-ISDN ATM adaptation layer type 2 specification," *Swiss Telecom PTT*, Sep. 1997.

[3] N. Kazuhiko and F. Hiroshi, "Evaluation of AAL-2 for low-bit-rate ATM voice communications," *NTT Review*, vol. 10, no. 1, pp. 72-80, Jan. 1998.

[4] "Voice over frame relay implementation agreement," *Frame Relay Forum Technical Committee*, FRF.11, May 1997.

[5] H. Nakamura, H. Tsuboya, M. Nakano, and A. Nakajima, "Applying ATM to mobile infrastructure networks," *IEEE Commun. Mag.*, vol. 36, no. 1, pp. 66-73, Jan. 1998.

[6] *Altera Data Book 1996*, Altera Corporation, USA, June 1996.

[7] P. J. Ashenden, *The Designer's Guide to VHDL*, Morgan Kaufmann, Apr. 1996.

[8] M. Braun, J. Friedlich, J. Lembert, and Grün,"Parallel CRC computation in FPGAs," in *FPL'96 Workshop on Field Programmable Logic and Applications*, Darmstadt, Germany, Sep. 1996.

[9] J. H. Baldwin, B. H. Bharucha, B. T. Doshi, S. Dravida, and S. Nanda, "AAL-2–A new ATM adaptation layer for small packet encapsulation and multiplexing," *Bell Labs Technical Journal*, pp. 111–131, Spring 1997.

Matti Tommiska was born in Valkeala, Finland on August 6, 1970. He received the M.Sc. degree in communication engineering from Helsinki University of Technology, Finland, in 1996 and the Licentiate of Science in Technology Lic.Sc.(Tech.) degree from the same university in 1998. He is currently working on his Ph.D. thesis evolving hardware aspects of telecommunication devices.

Mika V. Loukola was born in Espoo, Finland on April 12, 1973. He received the M.Sc. degree in communication engineering from Helsinki University of Technology, Finland, in 1996 and the Licentiate of Science in Technology Lic.Sc.(Tech.) degree from the same university in 1997. He is currently putting the finishing touches to the Ph.D. thesis on "Introducing Quality of Service To IP Packet Forwarding". He has written a number of papers and articles on those subjects.

Tero Koskivirta was born in Isojoki, Finland, on October 30, 1961. He received 1986 the M.Sc. degree in electrical engineering from Helsinki University of Technology, Finland, in 1986. He joined Nokia Data in 1986 and Nokia Research Center in 1990. His research interests are ATM protocols and signaling.