

Optimizing the Processing Performance of a Smart DMA Controller for LTE Terminals

David Szczesny, Sebastian Hessel, Shadi Traboulsi, Attila Bilgic

Institute for Integrated Systems, Ruhr-Universität Bochum

D-44780 Bochum, Germany

Email: {david.szczesny,sebastian.hessel,shadi.traboulsi,attila.bilgic}@is.rub.de

Abstract—In this paper we present an extended and optimized version of a smart Direct Memory Access (sDMA) controller supporting different on-the-fly protocol stack acceleration concepts for Long Term Evolution (LTE) mobile terminals. In addition to the downlink processing, we analyse different on-the-fly hardware acceleration modes for the uplink protocol stack processing in layer 2 (L2). Moreover, the system performance is further improved by adopting parallelization methods. The efficiency of on-the-fly hardware acceleration is proved by comparing the transport block processing times to those achieved with a conventional hardware accelerator. Therefore, a cycle approximate virtual prototype of a state-of-the-art mobile phone platform based on an ARM1176 processor is simulated at LTE-Advanced data rates of up to 1 Gbit/s. In uplink direction, we are able to reduce the complexity in the sDMA controller and simultaneously improve the processing performance in the mobile platform. This is realized by intelligent hardware/software partitioning and an optimized descriptor format. Furthermore, a significant optimization (up to 13 %) of the system performance in a mobile device is achieved by adopting parallelized on-the-fly hardware acceleration modes. We show how the sDMA controller clearly outperforms the traditional approach by reaching speedups of up to 35 % and 66 % for the transport block processing times in uplink and downlink directions, respectively.

I. INTRODUCTION

In 3GPP's Long Term Evolution (LTE) and in the next upcoming mobile communication standard LTE-Advanced, the peak data rates are increased up to 100 Mbit/s and even 1 Gbit/s, respectively. This general trend in wireless technologies is mainly driven by new services and features like video streaming or online gaming. Simultaneously growing algorithm complexity and decreasing latencies are challenging factors for mobile devices where computational resources and battery lifetime are strictly limited. Even though a lot of work has already been done regarding the physical layer [1], intensive investigations of higher protocol stack layers like the layer 2 (L2) are still required to support these high transfer rates in mobile terminals.

With efficient hardware/software partitioning concepts, that adopt dedicated hardware accelerators for the time critical tasks, the processing performance and the power consumption of an embedded system can be improved [2], [3]. Therefore, we presented already a promising approach for the LTE downlink protocol stack processing in [4]. It is based on a smart Direct Memory Access (sDMA) controller which significantly outperforms conventional hardware accelerators by applying on-the-fly hardware acceleration for the header

decoding and decryption. In this work, however, we investigate different on-the-fly hardware acceleration concepts for the uplink protocol stack processing. Moreover, by simulating LTE-Advanced transmission conditions of up to 1 Gbit/s, we show how the system performance of mobile terminals can be further increased with parallelization of the sDMA operations. A virtual prototyping approach is used for simulation and measurements of a mobile phone platform [5]–[8]. With its high simulation speeds and still satisfactory accuracy, it offers the necessary amount of abstraction for performance analyses on system level.

This paper is organized as follows: In section II the model of the mobile phone platform and the software stack are described. A detailed insight into the sDMA controller and the parallelization techniques is given in section III. The comparison of different hardware acceleration concepts for the protocol stack processing is presented in section IV. In section V the simulation setup and the results are presented leading to the conclusion in section VI.

II. SYSTEM ARCHITECTURE

The integration and evaluation of different hardware acceleration concepts for the LTE protocol stack is carried out in this work with a virtual prototype of a mobile phone. Its system architecture is depicted in Fig. 1 and consists of a cycle approximate model of a hardware platform which is simulated together with the executed software stack in the CoMET virtual prototyping environment provided by Synopsys [9].

A. Hardware Platform

The core of the hardware platform is an ARM1176 embedded processor [10] representing a common choice for state-of-the-art mobile devices [11]. It provides 64-bit and 32-bit AMBA AXI bus interfaces [12] connecting to the instruction and data buses and a peripheral bus, respectively. Different memory types are utilized in the hardware platform, a small on-chip memory with fast access times and a large off-chip memory with higher latencies [13]. Because of the high amount of data which needs to be transferred in LTE between the memory and the physical interface, the protocol stack is executed from the off-chip memory. In order to enable a realistic simulation of LTE transmissions for uplink (UL) and downlink (DL) directions in the closed virtual prototyping

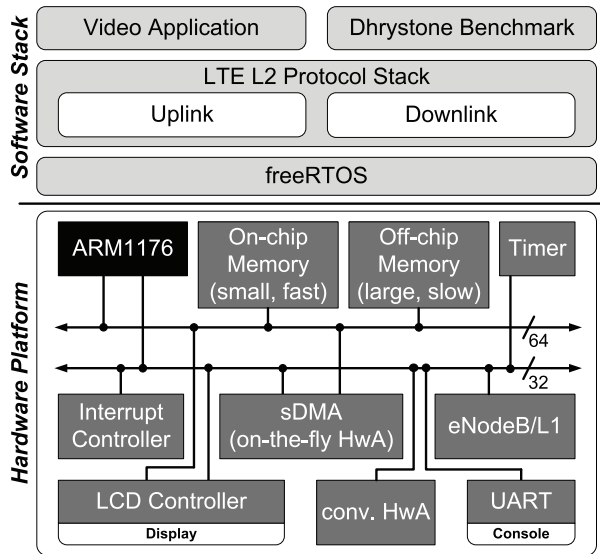


Fig. 1. System architecture of the ARM1176 based virtual prototype of a mobile phone showing the hardware platform and the software stack

environment, the physical layer and base station functionality are modeled by the configurable eNodeB/L1 peripheral. Thereby, transmitted or received data is generated or processed at different data rates and channel conditions on transport block level [14]. Moreover, the hardware platform contains a general purpose interrupt controller attached to all system interrupt lines, a timer which establishes a time base for the scheduler in the real-time operating system and two user interfaces: a UART connected to an emulated console for textual interaction and a display driven by an LCD controller for graphical outputs. The components in focus of this work are the hardware accelerators for the L2 protocol stack processing. On the one hand we use a stand-alone de-/ciphering unit (conv. HwA) supporting the encryption algorithms specified by the 3GPP [15] for LTE systems. It represents a conventional hardware acceleration approach with an internal data buffer and an interrupt signaling the processing completion [16]. On the other hand, an on-the-fly hardware acceleration concept is realized with the smart DMA (sDMA) controller. The de-/ciphering functionality is therefore integrated in the DMA controller and the DMA engine is extended by control and computational units for header decoding in downlink and descriptor processing in uplink directions.

B. Software Stack

In order to meet real-time requirements in protocol stack implementations, the underlying operating system should feature fast interrupt handling and low task switching latencies. Therefore, the software stack is based on the open source real-time operating system FreeRTOS™ in the current version 6.0.1 [17]. Its kernel supports tasks and co-routines whereas queues, semaphores and mutexes allow for inter task communication and synchronization. In our setup, the kernel is configured for preemptive mode enabling task interruption. This is mandatory

for the protocol stack execution where high priority tasks are activated for instance by incoming transport blocks. The LTE L2 protocol stack model implements the most complex and computational intensive functionality of the data plane [18]. Furthermore, the downlink and uplink subcomponents are subdivided each in only three different tasks corresponding to the sublayers Medium Access Control (MAC), Radio Link Control (RLC) and Packet Data Convergence Protocol (PDCP) specified by the 3GPP [19]. A lower number of tasks reduces the total task switching overhead in the real-time kernel. Video data is generated by a video application and is sent via the protocol stack to the emulated base station in uplink direction. Instead, received transport blocks containing video data are processed in downlink direction and forwarded to the video application which utilizes the LCD controller to play the video stream. In a realistic scenario, the protocol stack execution can be negatively affected by other applications running in parallel, for example by clearing the caches in the processor. We emulate such a behavior by executing a dhrystone benchmark permanently in parallel to all other tasks in the system.

III. SMART DMA (SDMA) CONTROLLER

A conventional DMA engine is combined in the sDMA controller with processing units for the LTE L2 protocol stack. This enables on-the-fly hardware acceleration applied directly during transport block transfers between the physical layer interface and the memory. In the following the extended sDMA engine and parallelization strategies for on-the-fly hardware acceleration are described in detail.

A. sDMA Engine

The key component of the sDMA controller is the extended sDMA data move engine depicted in Fig. 2. In the normal mode of operation with deactivated and bypassed hardware accelerators for a DMA channel, buffered read and write request are issued on the bus interfaces whereas the data available in the read buffer is moved directly to the write interface. This technique is used to perform processor independent copy operations between memory regions specified by source and destination addresses. Thus, a reduction of the CPU load and faster copy operations by employing burst transfers especially for big data sizes are achieved. Additionally, the sDMA controller can be configured for hardware acceleration accomplished during data transfers (on-the-fly) of LTE transport blocks from or to the physical layer interface in DL and UL directions, respectively.

On-the-fly accelerated processing in UL direction starts with reading descriptors for RLC protocol data units (PDUs) and PDCP PDUs, which are contained in a transport block, and storing them in the internal descriptor buffer. The sDMA controller is capable of reading the descriptors from a continuous memory region and processing descriptors organized in a linked list as well. In order to perform on-the-fly ciphering of the packets located in the transport block, the descriptor data is evaluated by the corresponding uplink processing unit which generates control information for the initialization and

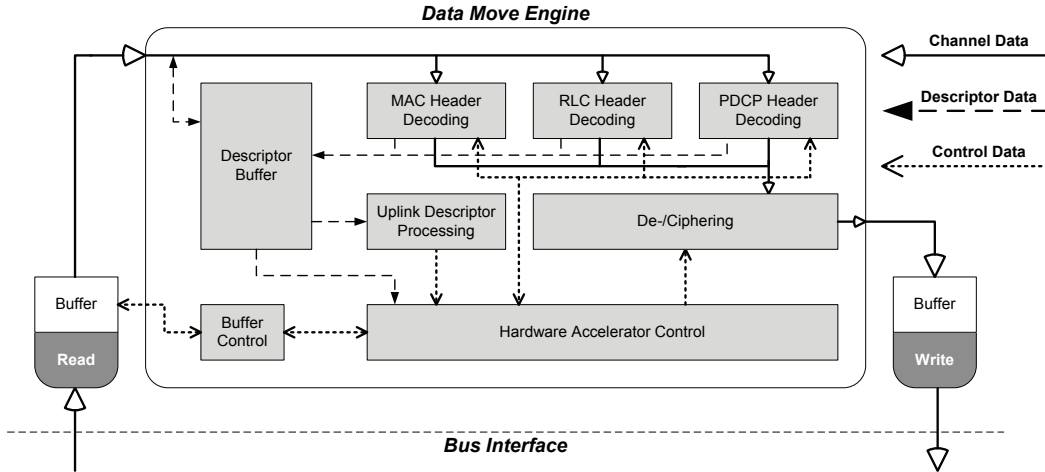


Fig. 2. Extended data move engine in the sDMA controller with uplink and downlink hardware accelerators and control units

activation of the de-/cipherng core. Every PDCP PDU has a unique sequence number and an associated hyper frame number required for individual initialization of the encryption algorithms. Furthermore, cipherng is not necessarily applied on entire PDCP packets which requires additional information in the form of an address offset referenced to the packet start address in the memory. A match between the current address offset calculated by the buffer control unit and obtained from the UL descriptor processing unit triggers a signal which activates the on-the-fly de-/cipherng unit. The encryption algorithms are then applied on all the data moved from the read to the write interface. In case of an empty read buffer the de-/cipherng unit is suspended and reactivated when data becomes available.

In contrast to the UL mode, the information required for on-the-fly hardware acceleration in DL direction needs to be extracted directly from the transport block data copied from the physical layer interface to the memory for further software processing by the protocol stack. Therefore, the header decoding units for the different L2 sublayers are activated starting with the MAC header decoder. The extracted information is stored in the descriptor buffer and triggers in combination with the information from the buffer control unit an enable signal for the RLC header decoder which in turn activates the following PDCP header decoder in the same way. An incomplete header in the read buffer which is identified by the amount of remaining data in the read buffer and the control information from the corresponding header decoding unit requires to halt the header processing and wait for the reactivation triggered by the next read burst. After successful completion of the PDCP header decoding, the de-/cipherng unit is initialized and performs on-the-fly decipherng for every not segmented PDCP PDU located in the transport block. Finally, the decoded header information from the descriptor buffer is copied after completion of the transport block transfer to preconfigured memory regions for each sublayer and further access by the protocol stack.

B. Parallelization

On architectural level, performance optimizations are carried out basically by minimizing the critical path and thus increasing the frequency. Furthermore, a higher data throughput can be achieved by pipelining and/or hardware parallelization. These optimization techniques only have direct impact on the timing of separate processing units like those inside of the sDMA controller. In fact, on system level this is modeled with parameters that are configured before simulation. However, system level simulations are mandatory for investigations of more abstract performance optimization approaches for peripherals whose processing times are influenced by the interaction with other components in the hardware platform. For instance, the timing for on-the-fly hardware acceleration with the sDMA controller depends among others on the read and write latencies from/to memory regions and on the current bus utilization.

The performance of on-the-fly accelerated transport block processing in both transmission directions can be increased by implementing separate bus master read and write interfaces and running them in parallel. Thereby, according to our setup a speedup can be achieved only by using different

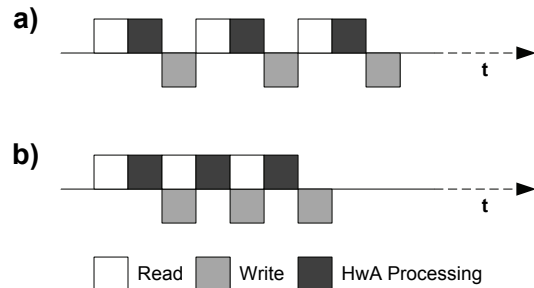


Fig. 3. Theoretical view on the timing characteristics during on-the-fly hardware acceleration with the sDMA controller: a) sequential mode with one bus master interface b) parallel mode with two bus master interfaces

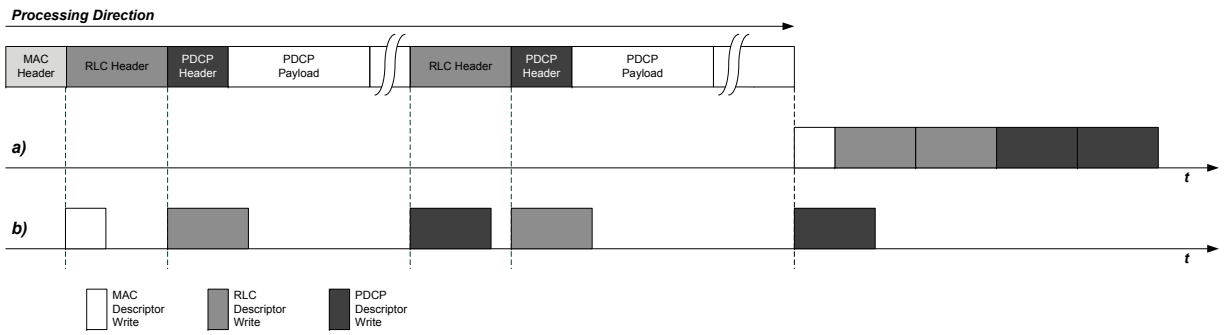


Fig. 4. Timing characteristics of the descriptor write operations during on-the-fly hardware acceleration with the sDMA controller in downlink direction: a) sequential mode where the descriptors are written after the transport block processing b) parallel mode with an additional bus master interface and descriptor write operations during transport block processing

busses connected to the physical layer interface and memories. Otherwise the concurrent read and write requests on the same bus would block each other which in turn represents the mode with one bus master interface. A theoretical comparison of the timing characteristics in the sequential mode with one bus master interface and the parallel mode by utilizing two bus master interfaces is illustrated in Fig. 3. We assume the same timings for the read and write bursts and the processing in the hardware accelerators. In the sequential mode, a write burst request is issued on the bus after completion of the read burst and the following processing in the hardware accelerators. The processing time is reduced in the parallel mode because of its capability to issue the write and the next read bursts simultaneously.

The timing required by the sDMA controller for on-the-fly hardware acceleration in DL direction can be further reduced by parallelizing the descriptor write operations (c.f. Fig. 4). Therefore, an additional bus master interface is necessary. In order to show the timing characteristics in sequential and parallel modes, an exemplary transport block, which comprises two RLC PDUs with an arbitrary number of PDCP PDUs, is used. It starts with a MAC header, followed by the RLC PDUs. Every RLC PDU consists of an RLC header and the payload which is represented by an arbitrary number of PDCP PDUs consisting of a header and a payload as well. When applying the sequential mode, the decoded header information in the form of descriptor data is written to the specified memory regions after the completion of the transport block processing. In the parallel mode, descriptor write operations are triggered directly after the corresponding header decoder finishes its computations. Because all PDCP PDUs contained in a RLC PDU are combined in one PDCP descriptor, the decoded header parameters are written to the memory after the processing completion of an entire RLC PDU.

IV. HARDWARE ACCELERATION CONCEPTS

Different hardware acceleration concepts for the LTE L2 protocol stack processing are evaluated in this work with regard to their performance measured on system level. The approaches are presented separately for the UL and the DL transmission directions.

A. Uplink

Transport blocks which are transmitted in UL direction over a wireless link are processed in the LTE L2 starting in the PDCP sublayer. Thereby, the payload of every PDCP PDU needs to be encrypted by applying the ciphering algorithms before the packets can be forwarded to the RLC subcomponent. All logical channel data from the RLC sublayer is combined in the MAC sublayer to a transport block which is then copied to the physical layer interface. When the conventional hardware accelerator is used, the payload of every PDCP PDU is moved by a DMA copy operation between the memory and the stand-alone de-/ciphering unit which encrypts the data (c.f. Fig. 5). In the on-the-fly hardware acceleration mode with the sDMA controller, the PDCP PDU payload is directly encrypted during the transport block transfer from the memory to the physical layer interface. Hence, the transport block data needs only to be copied once in contrast to the conventional approach. In order to perform on-the-fly hardware acceleration, necessary protocol stack information in the form of descriptors can be read and evaluated by the sDMA controller in two different ways. On the one hand, additional effort in the software stack is avoided by using the protocol stack descriptor format. The sDMA then processes descriptors organized in a linked list provided by the RLC and the PDCP sublayers. The drawbacks in the sDMA controller are a more sophisticated read mechanism for the linked lists and an overhead due to unused descriptor data. The protocol stack maintains a lot of different parameters in a packet descriptor which are not all required by the sDMA controller for on-the-fly hardware acceleration. In the second mode, an sDMA specific and optimized descriptor with removed redundancy is generated by the protocol stack in a continuous memory region. This additional complexity in the software stack is compensated in the sDMA controller with a simplified read mechanism and faster descriptor read operations.

B. Downlink

The comparison between the conventional and the on-the-fly hardware acceleration approaches in downlink direction is given in Fig. 6. Received LTE data is copied in the form

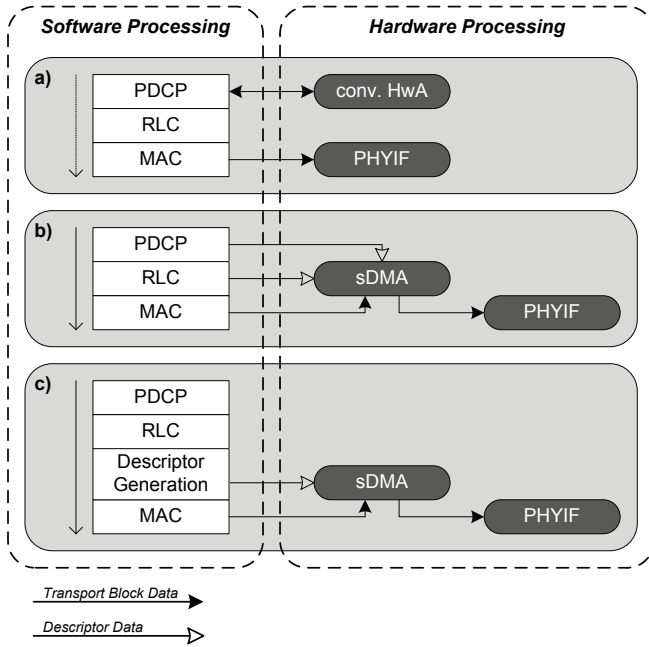


Fig. 5. Hardware acceleration concepts for the LTE L2 uplink: a) conventional with a stand-alone hardware accelerator b) on-the-fly with sDMA controller by using the protocol stack descriptor format c) on-the-fly with sDMA controller by using an optimized descriptor format

of transport blocks from the physical layer interface to the memory for processing by higher protocol stack layers. Before all PDCP PDUs located in the transport block can be extracted and deciphered in the PDCP sublayer, the MAC, RLC and PDCP headers are sequentially decoded and evaluated. Deciphering with the conventional hardware accelerator is accomplished on PDCP level where encrypted data is copied from the memory to the stand-alone de-/ciphering unit. After the completion signaled by an interrupt, the decrypted data is transferred back to the memory. In contrast, on-the-fly hardware acceleration including deciphering takes place directly during copying the transport block data from the physical layer interface to the memory. Furthermore, necessary header information extracted by the header decoding units in the sDMA controller is written in the protocol stack descriptor format to the specified memory regions for access by the MAC, RLC and PDCP sublayers. Consequently, the processing complexity and execution times in the protocol stack are further reduced by omitting the header decoding functionality in software.

V. RESULTS

All parameters controlling the system behavior of the mobile phone platform are set in a global configuration file in the virtual prototyping environment before simulation and measurement. We use clock frequencies of 500 MHz and 200 Mhz on the ARM1176 processor and the bus system, respectively. According to [20], this is a reasonable setting for LTE mobile phones. Moreover, the core is configured with data and instruction cache sizes of 32 kB. Investigations

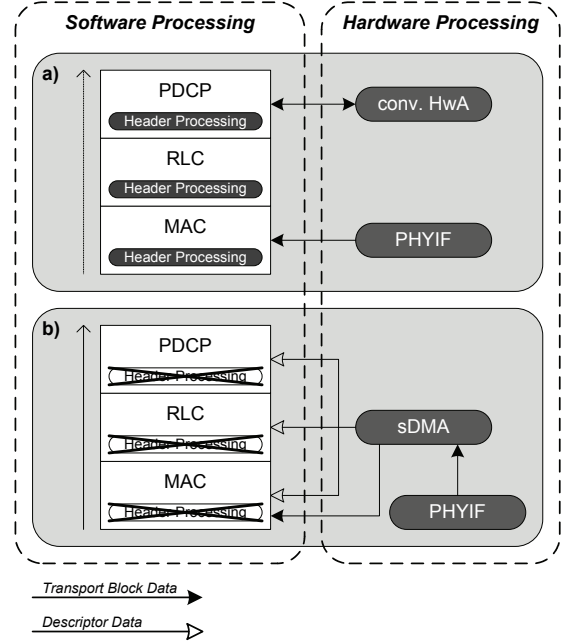


Fig. 6. Hardware acceleration concepts for the LTE L2 downlink: a) conventional with a stand-alone hardware accelerator b) on-the-fly with sDMA controller

have shown, that an increase of the cache sizes leads only to marginal performance improvements [14]. The stand-alone and the integrated de-/ciphering unit in the sDMA controller use both an equal timing configuration of 4 ns per byte for encryption and decryption. This is the maximum performance value derived from hardware implementations presented in [21]. Additionally, the header decoding units in the sDMA controller work with a timing of one cycle per byte. This assumption is justified by the fact that the extraction of header information from dedicated bit fields does not require intensive calculations. The processing times in the protocol stack are measured for different transport block sizes generated by the eNodeB/L1 peripheral (DL) and by the protocol stack (UL). With the transmission time interval of 1 ms in LTE, the variation between 100 kbit and 1 Mbit corresponds to data rates of 100 Mbit/s and 1 Gbit/s, respectively. All processing times presented in this work are derived by calculating the average value from 1000 measurements.

The average processing times per transport block of the LTE L2 UL for different hardware acceleration concepts are depicted in Fig. 7. The conventional hardware acceleration approach (conv. HwA) is compared to different UL on-the-fly hardware acceleration modes in the sDMA controller by using the protocol stack descriptor format (sDMA UL1) and an optimized descriptor format (sDMA UL2). Additionally, the sDMA controller operates first in a sequential and second in a parallel mode with two bus master interfaces. As expected, the processing times are strongly increasing with the data rates. It becomes clear that all on-the-fly hardware acceleration concepts are significantly outperforming the conventional

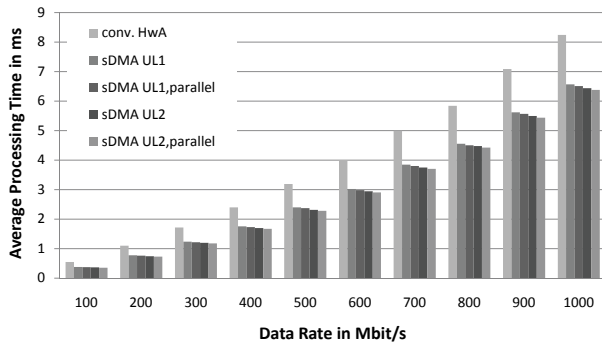


Fig. 7. Average processing times per transport block of the LTE L2 UL protocol stack model for different hardware acceleration modes

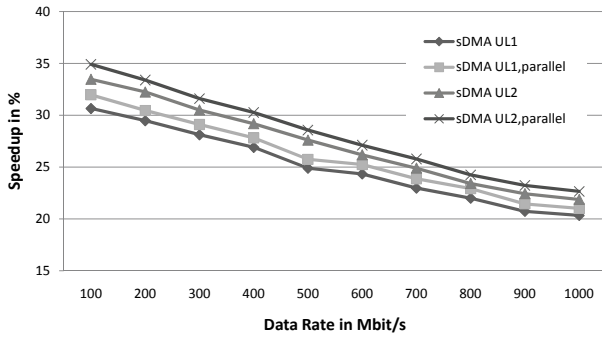


Fig. 8. Relative speedup of the LTE L2 UL processing for different hardware acceleration modes in the sDMA controller in comparison to a conventional hardware acceleration concept

hardware accelerator for all transmission conditions. A more detailed diagram showing the relative speedups achieved with on-the-fly hardware acceleration in UL direction compared to the conventional approach is given in Fig. 8. The speedup is decreasing with higher data rates for all sDMA configurations and ranges at most between 27 % and 35 % (sDMA UL2, parallel). The reason is that the descriptor processing complexity grows faster with increased data rates than the processing of the transport block data resulting in a reduced performance gain. Compared to the sDMA UL1 hardware acceleration concept, the sDMA UL2 approach is up to 3 % faster in the sequential as well as in the parallel mode.

Even higher speedups are achieved by on-the-fly hardware acceleration in DL direction compared to the conventional approach (c.f. Fig. 9 and Fig. 10). In addition to the sequential mode, the sDMA accelerated processing times are measured for the parallel mode with two bus master interfaces (parallel1) and the full parallelization with an extra bus master interface and concurrently running descriptor write operations (parallel2). The header information is directly decoded in the sDMA controller in contrast to the UL mode where first descriptors generated by the protocol stack need to be read and evaluated. Hence, a higher maximum performance gain of about 66 % (sDMA DL, parallel2) is reached for DL on-the-fly hardware acceleration whereas the speedup is increasing with growing data rates.

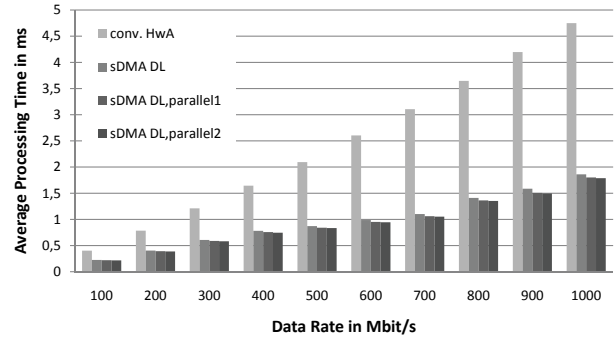


Fig. 9. Average processing times per transport block of the LTE L2 DL protocol stack model for different hardware acceleration modes

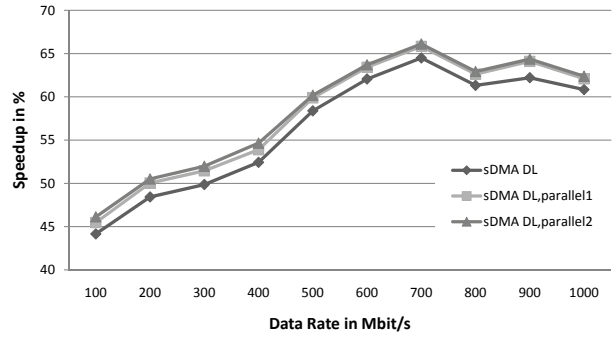


Fig. 10. Relative speedup of the LTE L2 DL processing for different hardware acceleration modes in the sDMA controller in comparison to a conventional hardware acceleration concept

In order to analyze the effectiveness of the parallelization techniques in comparison to the sequential modes in the sDMA controller, the relative performance improvement of the processing times during on-the-fly hardware acceleration is depicted in Fig. 11. By implementing an additional bus master interface in the sDMA controller for parallel read and write, the processing time can be reduced by up to 9 % and up to 10 % for the UL and DL modes, respectively. At higher data rates, with more effort for the descriptor read and processing where parallelization has no effect in UL direction, the speedup is degraded to approximately 3 %. In contrast, the

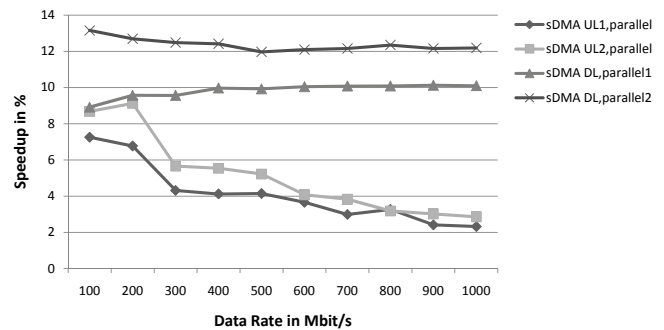


Fig. 11. Relative speedup of the sDMA processing times achieved by parallelization concepts for on-the-fly hardware acceleration in uplink and downlink directions

DL speedup is almost constant and can be further improved (13 %) by applying the parallel descriptor write mode.

VI. CONCLUSION

In this paper we present different on-the-fly hardware acceleration concepts for uplink (UL) and downlink (DL) LTE protocol stack processing based on a smart DMA (sDMA) controller for mobile terminals. Moreover, by applying parallelization techniques, we show how the efficiency of the sDMA controller is improved. We investigate the performance on system level by comparing the processing times to those achieved with a conventional hardware accelerator. Therefore, a cycle approximate virtual prototype of a mobile phone consisting of an emulated hardware platform and the software stack is simulated with LTE-Advanced data rates of up to 1 Gbit/s in both transmission directions. Best performance is offered in UL direction with the sDMA controller in the optimized descriptor mode. More computational effort in the protocol stack is compensated with less complexity and thus faster processing in the hardware accelerator. Consequently, this is the preferred mode for on-the-fly hardware acceleration in UL direction. Furthermore, by incrementing the number of bus interfaces and parallelizing the sDMA transfers we reduce the sDMA processing times by up to 13 %. Considering a full utilization of the sDMA controller also by other applications in the mobile terminal issuing conventional DMA transfers, the whole system performance is equally improved. On-the-fly hardware acceleration has a clear benefit compared to the conventional hardware acceleration approach with maximum speedups of up to 35 % and 66 % in UL and DL directions, respectively. Simultaneously, the adoption of the sDMA controller significantly reduces the number of data transfers leading to a reduced utilization of the bus architecture. Since the overall power consumption in embedded systems considerably depends on the bus communication [22], we assume that a noticeable decrease can be reached with our on-the-fly hardware acceleration approach. We show that the sDMA controller with its high performance and relative low complexity represents an efficient hardware accelerator for LTE terminals.

ACKNOWLEDGMENT

The authors acknowledge the excellent cooperation with all project partners within the EASY-C project and the support by the German Federal Ministry of Science and Education (BMBF). Further information is available on the project website: <http://www.easy-c.de>.

REFERENCES

[1] J. Berkmann, C. Carbonelli, F. Dietrich, C. Drewes and Wen Xu, "On 3G LTE Terminal Implementation - Standard, Algorithms, Complexities and Challenges", in *International Wireless Communications and Mobile Computing Conference 2008 (IWCMC '08)*, Crete Island, Greece, pp. 970-975, August 2008

[2] Greg Stitt, Roman Lysecky and Frank Vahid, "Dynamic Hardware/Software Partitioning: A First Approach", in *Proceedings of the 40th Design Automation Conference (DAC 2003)*, Anaheim, California, USA, pp. 250-255, June 2003

[3] M. Ouellette, D. Connors, "Analysis of Hardware Acceleration in Reconfigurable Embedded Systems", in *19th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2005)*, Denver, Colorado, USA, pp. 168a-168a, April 2005

[4] D. Szczesny, S. Hessel, F. Bruns and A. Bilgic, "'On-the-fly Hardware Acceleration for Protocol Stack Processing in Next Generation Mobile Devices'", in *7th International Conference on Hardware/Software Code-sign and System Synthesis (CODES+ISSS 2009)*, Grenoble, France, pp. 155-162, October 2009

[5] J. Cockx, "Efficient Modeling of Preemption in a Virtual Prototype", in *11th International Workshop on Rapid System Prototyping (RSP 2000)*, Paris, France, pp. 14-19, June 2000

[6] S. Schliecker, J.-C. Braam, S. Stein and M. Schnieringer, "White Paper: Software Driven Embedded Systems Design"

[7] T. Eckart and M. Schnieringer, "Development and Verification of Embedded Firmware using Virtual System Prototypes", in *International Symposium on System-on-Chip (SoC 2006)*, Tampere, Finland, pp. 1-1, November 2006

[8] M. Brandenburg, A. Schollhorn, S. Heinen, J. Eckmüller and T. Eckart, "From Algorithm to First 3.5G Call in Record Time - A Novel System Design Approach Based on Virtual Prototyping and its Consequences for Interdisciplinary System Design Teams", in *Design, Automation & Test Conference (DATE 2007)*, Nice, France, pp. 1-3, April 2007

[9] Synopsys, Inc., <http://www.synopsys.com>

[10] *ARM1176JZF-S Processor Technical Reference Manual*, ARM Limited, Lit.-Nr.: ARM DDI 0301F, 2008

[11] S. Hessel, F. Bruns, A. Bilgic, A. Lackorzynski, H. Härtig and J. Hausner, "Acceleration of the L4/Fiasco Microkernel Using Scratchpad Memory", in *International Workshop on Virtualization in Mobile Computing (MobiVirt 2008)*, Breckenridge, pp. 6-10, USA, June 2008.

[12] *AMBA AXI Protocol*, ARM Limited, Lit.-Nr.: ARM IHI 0022B, 2004, <http://www.arm.com>

[13] P. R. Panda, N. D. Dutt, and A. Nicolau, "'On-chip vs. off-chip memory: the data partitioning problem in embedded processor-based systems'" *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Volume 5, Issue 3, July 2000

[14] D. Szczesny, A. Showk, S. Hessel, U. Hildebrand, V. Frascolla and A. Bilgic, "Performance Analysis of LTE Protocol Processing on an ARM based Mobile Platform", accepted for *11th International Symposium on System-on-Chip (SoC 2009)*, Tampere, Finland, October 2009

[15] *3GPP System Architecture Evolution (SAE): Security Architecture*, 3GPP Std. TS 33.401, Rev. 8.2.0, Dec. 2008.

[16] Olli Silven and Kari Jyrkkä, "Observations on Power-Efficiency Trends in Mobile Communication Devices", in *EURASIP Journal on Embedded Systems*, Volume 2007, ISSN:1687-3955, January 2007.

[17] *FreeRTOS™*, <http://www.freertos.org>

[18] D. Szczesny, A. Showk, S. Hessel, U. Hildebrand, V. Frascolla and A. Bilgic, "'Joint Uplink and Downlink Performance Profiling of LTE Protocol Processing on a Mobile Platform'", to appear in *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 2010

[19] *The 3rd Generation Partnership Project (3GPP)*, <http://www.3gpp.org>

[20] van Berkel, C.H., "Multi-core for mobile phones," in *Design, Automation & Test in Europe Conference (DATE '09)*, Nice, France, pp.1260-1265, April 2009

[21] S. Hessel, D. Szczesny, N. Lohmann, J. Hausner and A. Bilgic, "Implementation and Benchmarking of Hardware Accelerators for Ciphering in LTE Terminals", submitted in *IEEE Global Communications Conference (GLOBECOM 2009)*, Honolulu, Hawaii, USA, November 2009.

[22] C. Talarico, J.W. Rozenblit, V. Malhotra and A. Stritter, "A New Framework for Power Estimation of Embedded Systems", in *Computer Volume 38*, pp. 71-78, February 2005