

Performance Analysis of LTE Protocol Processing on an ARM based Mobile Platform

David Szczesny, Anas Showk, Sebastian Hessel, Attila Bilgic

Institute for Integrated Systems, Ruhr-Universität Bochum

D-44780 Bochum, Germany

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

Uwe Hildebrand, Valerio Frascolla

Comneon GmbH

D-90449 Nürnberg, Germany

Email: {u.hildebrand,v.frascolla}@comneon.com

Abstract—In this paper we present detailed profiling results and identify the time critical algorithms of the Long Term Evolution (LTE) layer 2 (L2) protocol processing on an ARM based mobile hardware platform. Furthermore, we investigate the applicability of a single ARM processor combined with a traditional hardware acceleration concept for the significantly increased computational demands in LTE and future mobile devices. A virtual prototyping approach is adopted in order to simulate a state-of-the-art mobile phone platform which is based on an ARM1176 core. Moreover a physical layer and base station emulator is implemented that allows for protocol investigations on transport block level at different transmission conditions. By simulating LTE data rates of 100 Mbit/s and beyond, we measure the execution times in a protocol stack model which is compliant to 3GPP Rel.8 specifications and comprises the most processing intensive downlink (DL) part of the LTE L2 data plane. We show that the computing power of a single embedded processor at reasonable clock frequencies is not enough to cope with the L2 requirements of next generation mobile devices. Thereby, Robust Header Compression (ROHC) processing is identified as the major time critical software algorithm, demanding half of the entire L2 DL execution time. Finally, we illustrate that a conventional hardware acceleration approach for the encryption algorithms fails to offer the performance required by LTE and future mobile phones.

I. INTRODUCTION

Next generation mobile communication systems, like Long Term Evolution (LTE) and beyond provide increased data rates at simultaneously reduced latency to account for new features and services like video streaming or online gaming over wireless links. Consequently, the growing protocol stack complexity, associated with rising processing demands, is a huge challenge for the system architecture of mobile devices where computational power and battery lifetime are limited. An efficient hardware/software partitioning is generally crucial in embedded systems [1] and especially for LTE and future mobile platforms where real time requirements have to be met.

Up to now, researchers and developers are focussing on the algorithms located in the physical layer [2]. But the computational demands of higher protocol stack layers, like layer 2 (L2), are also heavily increasing with the strongly growing data rates and need extensive investigations as well. A detailed profiling and identification of time critical algorithms are mandatory for any effective hardware/software partitioning concept. Since LTE specifications have been practically finalized recently, profiling in such early phase requires ap-

propriate simulation and measurement methods, which allow for a communication between a base station and the analyzed mobile terminal in a fully controlled and closed environment.

In this paper we analyse the performance of LTE L2 protocol processing on an ARM based mobile platform by applying virtual prototyping, a promising hardware/software co-design methodology for embedded systems [3] [4]. An ARM1176 processor based state-of-the-art mobile hardware architecture is implemented providing an eNodeB/L1 base station and physical layer emulator that allows for protocol analysis on transport block level in L2 at different transmission conditions associated with different computational demands on the terminal side. In several simulation scenarios, LTE data rates of 100 Mbit/s and beyond are processed in a protocol stack model executed in the hardware platform representing the most complex part of the LTE L2 DL data plane.

This paper is organized as follows: Section II describes the virtual hardware platform, consisting of the processor architecture with the eNodeB/L1 peripheral for parametrizable LTE and beyond data generation. An insight to the LTE protocol stack model and an introduction to the freeRTOS™ real time operating system are provided in section III. Section IV briefly explains the profiling workflow used to obtain simulation results, which are presented in section V. Finally the paper is concluded in section VI.

II. VIRTUAL HARDWARE PLATFORM

A virtual hardware prototype is used in this work for investigations and execution of an LTE protocol stack. Implementation, simulation and analysis of such a software model of a system on chip (SoC) are carried out using tools provided by VaST Systems Technology Corporation [5].

A. Processor Architecture

The processor architecture is a virtual prototype of a state-of-the-art mobile phone platform. An excerpt of it, with components in focus of this work, is illustrated in Fig. 1. The platform is based on an ARM1176 embedded processor [6] which provides AMBA AXI bus interface ports. Detailed information about this bus protocol can be found in [7]. Fast internal memory and external memory with higher read and write latencies are connected to the processor via a 64-bit instruction and data bus. In addition, a 32-bit peripheral bus

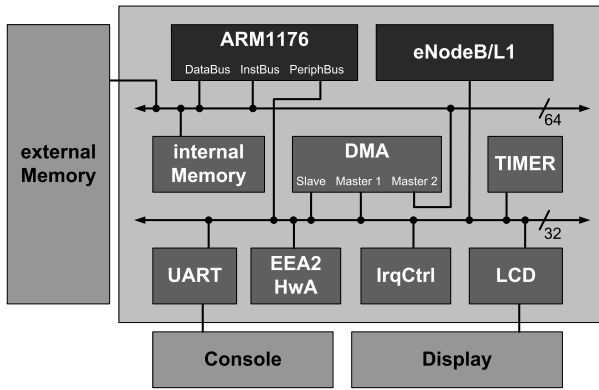


Fig. 1. Virtual Prototype of an ARM1176 based Mobile Hardware Platform

connects several devices to the core. All platform interrupts are combined and handled in an interrupt controller. Operating system interrupts are created by the timer. The eNodeB/L1 peripheral acts as a physical layer and base station emulator, generating LTE transport blocks. Each transport block is copied with a Direct Memory Access (DMA) controller to the external memory for protocol stack processing. The AES (EEA2) encryption algorithm used in LTE and specified by the 3GPP in [8], is supported in the hardware platform and in the protocol stack by the EEA2 hardware accelerator which represents a conventional hardware acceleration concept [9]. Furthermore, two user interfaces are integrated in the platform: A Universal Asynchronous Receiver Transmitter (UART) connected to a console and an LCD controller with an emulated display. During the simulation of LTE communication, a video is transmitted from eNodeB/L1 peripheral to the mobile platform and displayed by using the LCD controller.

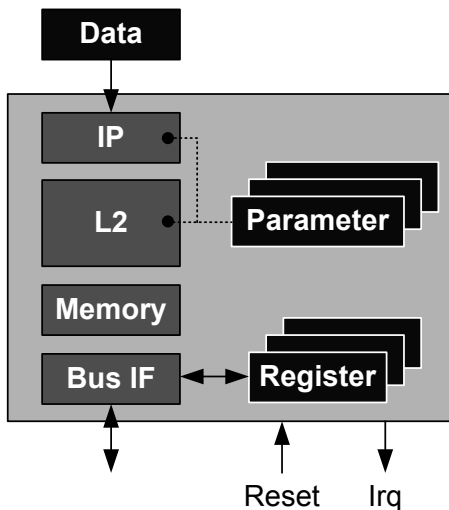


Fig. 2. eNodeB/L1 Physical Layer and Base Station Emulator for Transport Block Generation

B. eNodeB/L1 Peripheral

The LTE physical layer and a base station are emulated by the eNodeB/L1 peripheral that allows for LTE protocol investigations on transport block level in the closed virtual prototyping environment. Implementation and integration in the hardware platform is carried out using the C programming language and Architectural Modeling Programming Interface (AMPI), a proprietary C library for hardware modeling, provided by VaST.

An overview of the eNodeB/L1 peripheral is depicted in Fig. 2. It comprises LTE transport block generation units, IP and L2, with a file interface for data input, internal memory for transport block buffering and a bus interface for register and memory access. In addition to the reset port, an interrupt (Irq) port is implemented which signals the completion of each created transport block available in the internal memory. In order to account for different transmission conditions associated with different processing demands on mobile terminal side, the transport block generator is configured via parameters before simulation.

The data flow in the eNodeB/L1 peripheral and the impact of controlling parameters are illustrated in Fig. 3. IP packets with preconfigured sizes are created in the IP layer by reading video data from a file and adding an IP header. After that the IP packets are forwarded to the LTE L2 unit that generates 3GPP Rel.8 compliant transport blocks (see [10]–[13]) and stores them in the internal peripheral memory. The L2 unit is subdivided into three sublayers in the following processing order: Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC) and Medium Access Control (MAC). The L2 processing starts in the PDCP sublayer. Before adding PDCP headers to IP packets, the IP headers are compressed with

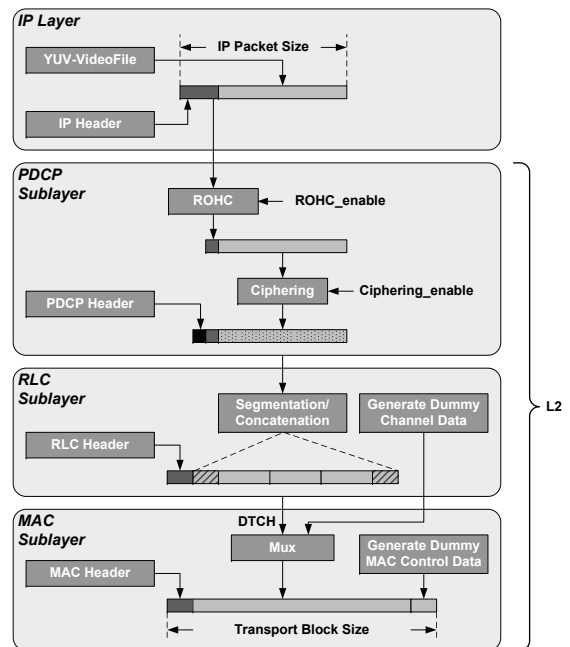


Fig. 3. Data Flow and Parameters in the eNodeB/L1 Peripheral

the Robust Header Compression (ROHC) and the IP packet payload is encrypted by applying the EEA2 ciphering algorithm. Both algorithms can be de-/activated via parameters. After the PDCP processing, PDCP protocol data units (PDUs) are forwarded to the RLC sublayer. The data transfer is carried out on the Dedicated Traffic Channel (DTCH), whereas the data size of each RLC PDU is typically determined by the physical layer transmission conditions. In our setup the RLC PDU size is simply 100 bytes smaller than the complete transport block size. The main goal of this work is the profiling of the data plane in LTE L2 and consequently we are not considering the physical layer and any physical channel information. Each RLC PDU is filled with PDCP PDUs or PDCP PDU segments in the segmentation/concatenation unit. Segments occur either at the beginning and/or at the end of RLC PDUs. After adding an RLC header, dummy data for other logical channels is generated in order to increase the header processing complexity in the MAC sublayer and hence providing a more realistic scenario. All logical channel data is multiplexed to the transport block payload in the MAC sublayer. Furthermore, dummy MAC control data like the timing advance is appended to the end of the transport block and a MAC header is added to finalize the transport block generation. In addition to parameters for the transport block size and the transmission time interval (TTI) representing the transport block generation period, a reordering buffer is used to account for packet loss and retransmission in wireless links. The reordering buffer size is specified by a reordering depth parameter. Generated transport blocks are copied in reverse order from the reordering buffer to the internal eNodeB/L1 peripheral memory for software access.

III. SOFTWARE STACK

The software stack executed in the previously presented hardware platform consists of an LTE protocol stack model that is implemented on top of the freeRTOS™ real time operating system.

A. Protocol Stack Model

The protocol stack model represents the most complex and therefore execution time critical part of the LTE L2 downlink (DL) data plane. Thereby, mainly the inverse functionality of the transport block generator in the eNodeB/L1 peripheral is implemented. Due to a relatively low computational effort, the control plane is not considered in this work.

Three sublayers form the LTE L2 data plane: MAC, RLC and PDCP (c.f. Fig. 4). Received transport blocks are copied to the external memory with the DMA controller. Afterwards a signal with the start address of the transport block (MAC PDU) triggers the MAC protocol processing. Each MAC PDU comprises several MAC service data units (SDUs) which are transferred after MAC completion to the RLC sublayer. Here MAC SDUs are called RLC PDUs which similarly contain some RLC SDUs that represent PDCP PDUs in the PDCP sublayer. In the last stage of the LTE L2 data plane, IP packets (PDCP SDUs) are obtained after processing in the

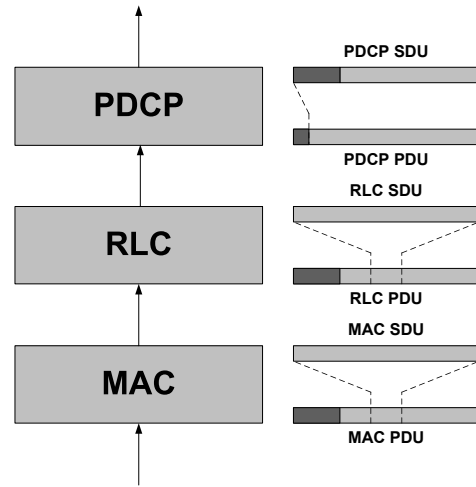


Fig. 4. Overview of the LTE Protocol Stack Model (L2 DL)

PDCP sublayer and forwarded to higher protocol layers. Input and output to the protocol stack model and communication between the sublayers within the protocol stack model are realized by pointers. This avoids inefficient copying of large memory blocks which significantly reduces the execution time. In the following, the implementation of each sublayer is described in detail.

1) *MAC*: In the MAC sublayer, MAC SDU information in form of data lengths and channel identification numbers is extracted during header processing applied on MAC PDUs. Additionally, the address offsets identifying the MAC SDU positions in the transport block are calculated. Based on this data, each MAC SDU is assigned to the corresponding logical channel signal in the demultiplexing module and forwarded to the RLC sublayer (c.f. Fig. 5).

2) *RLC*: Three different modes of operation are provided by the RLC sublayer: Transparent Mode (TM), Unacknowledged Mode (UM) and Acknowledged Mode (AM) (c.f. Fig. 6). In the TM mode PDUs are directly forwarded to the PDCP sublayer without any processing in the RLC sublayer in order to achieve low latencies on the Broadcast Channel (BCCH), the Common Control Channel (CCCH) and the Paging Control Channel (PCCH). These dummy channel information generated in the eNodeB/L1 peripheral

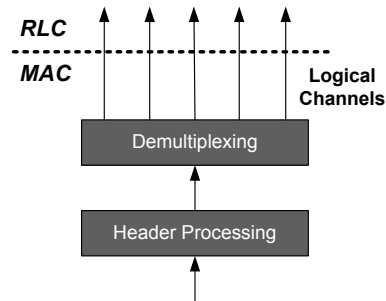


Fig. 5. Implemented functionality of the MAC sublayer

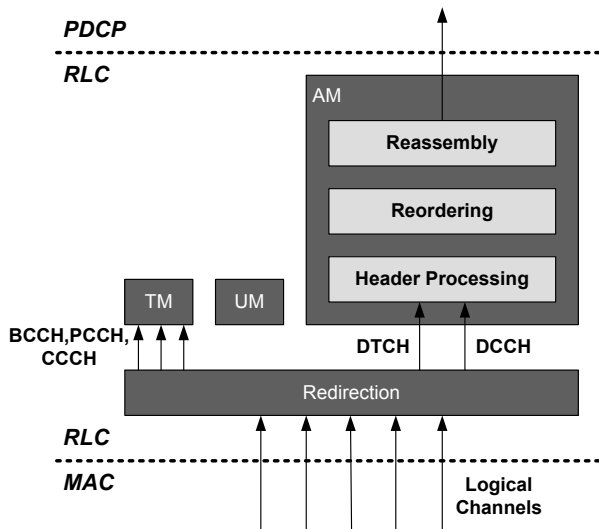


Fig. 6. Implemented functionality of the RLC sublayer

is just discarded in the TM entity in the protocol stack model. Furthermore the UM and AM mode is specified by the 3GPP for the DTCH channel. Besides additional features like RLC packet resegmentation support in the AM mode, the main difference between both modes is the existence of packet acknowledgement. Because of higher complexity and accordingly higher impact on the processing demand on mobile terminal side, only the AM mode is implemented in the protocol stack model. Therefore the DTCH and the Dedicated Control Channel (DCCH) are redirected to the AM entity, whereas the latter contains dummy control information that is discarded. Dynamic memory allocation provided by the operating system is used to store the decoded header data for each RLC PDU and all included RLC SDUs in linked lists for further processing in the protocol stack model. Based on the sequence numbers of the received RLC PDUs, the following reordering is carried out with a simple insertion sort algorithm. Finally, segments of RLC SDUs need to be reassembled in the corresponding entity before being transmitted to the PDCP sublayer. For that purpose, the DMA controller is used to copy the appropriate segments to one continuous memory region representing complete RLC SDUs.

3) *PDCP*: Each received PDCP PDU is processed in the PDCP sublayer in the following order: Header processing, deciphering and ROHC decompression (c.f. Fig. 7). After decoding the header information, the encrypted payload of each PDCP PDU is deciphered using the EEA2 hardware accelerator. Software implementations of encryption algorithms are highly inefficient compared to their hardware counterparts so that hardware acceleration is already used in state-of-the-art mobile platforms. Because of increasing data rates and algorithm complexity in future mobile devices, only hardware supported encryption is considered in this work. In addition to the key and the PDCP PDU header information, a hyper frame number (HFN) is maintained in the protocol stack model needed for a correct decryption of the received data. In order

to decipher the payload of a PDCP PDU, the data is copied to the EEA2 hardware accelerator by using a DMA controller and the decryption is started after configuring appropriate registers. An interrupt signals the completion and triggers the reverse DMA copy of the deciphered data to the external memory. For measurements of the pure software performance, the deciphering can be deactivated before compilation. As the last computing step in the PDCP sublayer, the ROHC decompression is applied on the compressed IP header. A modified and optimized version of the free implementation of ROHC available at [14] is therefore integrated in the protocol stack model. One major modification is the separation between the decompressed IP header and the payload. This approach avoids the inefficient creation of a continuous IP packet in the memory. Instead of that, memory is allocated only for the decompressed header whereas the payload remains in the original memory region. The extracted IP packets are forwarded to a very simple model of an IP layer, where only the checksum and the destination address are validated. Afterwards, the received video data is displayed using the LCD controller.

B. *freeRTOS*[™]

FreeRTOS[™] is an open-source real time operating system that supports tasks and co-routines. Inter task communication and synchronization is established via queues, semaphores and mutexes. Furthermore, the kernel can be configured for preemptive, cooperative or hybrid mode. More detailed information can be found in [15]. A real time operating system is mainly characterized by low interrupt and task switching latencies required by real time applications like an LTE protocol stack. The protocol stack model in this work is subdivided into 17 tasks with different priorities implemented on top of freeRTOS[™]. The kernel runs in the preemptive mode enabling task activation by receiving signals on queues. For instance a signal from the eNodeB/L1 interrupt handler activates a high priority task for copying the transport blocks to the external memory. After completion, a message with the pointer to the transport block invokes the next task in the MAC sublayer and

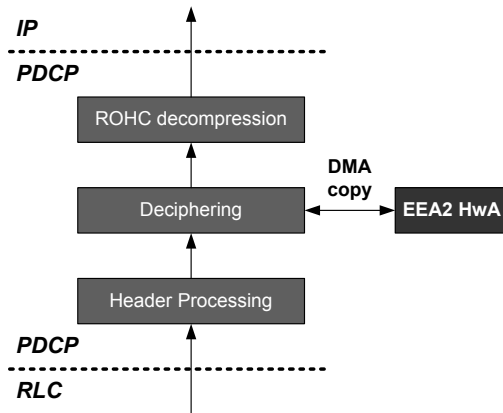


Fig. 7. Implemented functionality of the PDCP sublayer

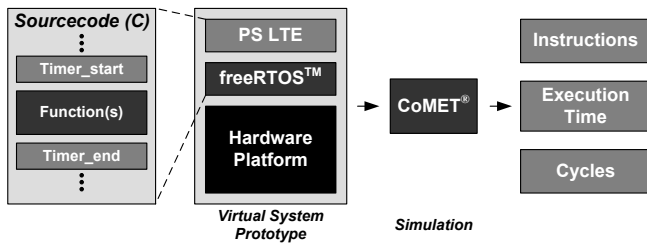


Fig. 8. Profiling Workflow of a Virtual Mobile Platform using CoMET®

so forth.

IV. PROFILING

A timing analysis of software execution is called performance profiling. The resulting identification of execution time critical algorithms is mandatory for an effective hardware/software partitioning in mobile embedded systems where real time constraints have to be met. The virtual system prototype of the mobile platform is profiled with the CoMET® engineering environment from VaST (c.f. Fig. 8). VaST’s API provides an interface from the software executed on a virtual processor architecture to the simulator which allows for exact timing measurements. In order to measure the execution time of determined parts in the protocol stack, timer functions are placed directly in the C code enclosing the dedicated functionality. In addition we use a Tool command language (Tcl) script, starting simulations with different parameters and extracting the timer values from the CoMET® output files afterwards. The evaluation is performed automatically by calculating the minimum, the average and the maximum values for the execution time, the number of instructions and clock cycles of target functions.

V. RESULTS

The platform is configured via parameters in the virtual system prototype before simulation and measurement. The ARM1176 processor clock and the bus clocks are set to 450 MHz and 200 MHz, respectively. This represents high clock frequencies compared to the state-of-the-art mobile platform presented in [16]. Furthermore, the processing time per byte of the EEA2 hardware accelerator for the deciphering is set to 10 ns according to realistic timing of the hardware implementations presented in [17]. In the first setup, the protocol stack model processes an LTE DL data rate of 100 Mbit/s generated by the eNodeB/L1 peripheral with a transport block size of 100 kbits and a transmission time interval (TTI) configuration of 1 ms. The transport block size and the IP packet size are varied in the second and third measurement scenario, respectively. In order to account for the packet loss and the retransmission in wireless links, the reordering depth parameter is used in the eNodeB/L1 peripheral. It is configured with a value of two, resulting in a sequence order of 2, 1, 4, 3, 6, 5, ... at transport block generation, leading to a higher processing complexity in the LTE protocol stack. The software stack is compiled with a

GCC version 4.3.2 cross compiler at the highest optimization level (-o3).

In this work we measure the average execution times of the L2 DL in the LTE protocol stack model by simulating the system for 1 sec. Because of operating system functionality between the sublayers, the average value for the entire L2 is not represented by adding the average values of all its subcomponents. The measurements are accomplished in two scenarios: deciphering disabled or enabled. The former is used to present profiling results of pure software execution without hardware support for the deciphering in the EEA2 peripheral. Execution time measurements of the complete data plane of the LTE DL protocol stack model are obtained in the deciphering enabled mode. Moreover, the system is simulated at different data and instruction cache sizes (8, 16, 32 and 64 kB) for the ARM1176 processor.

The average execution times of the LTE L2 DL protocol stack model in the deciphering disabled scenario are illustrated in Fig. 9. The instruction and data cache sizes are varied between 8 kB and 64 kB by doubling the cache size for each simulation. Additionally, the average execution time per transport block is measured for the entire L2 DL data plane and the sublayers. By increasing the data and instruction cache sizes to 64 kB, the average execution time per transport block of the whole L2 DL is reduced from 463 μ s to 355 μ s resulting in a speedup of approximately 23 %. Increasing the cache sizes beyond 32 kB only leads to a marginal performance improvement of about 1 %. With regard to the LTE requirements, where up to two transport blocks are processed per 1 ms, the L2 DL data plane processing demands already about 70 % of the available execution time of 0.5 ms. Taking into consideration a complete protocol stack implementation, additionally including the LTE L2 uplink and upper protocol stack layers, it becomes clear that a single ARM core at even higher frequencies is insufficient for LTE terminals. The relative distribution of the average execution time among the L2 DL subcomponents and of the functions within each subcomponent is shown in Fig. 10. The computational power of the L2 DL is mainly consumed by the PDCP sublayer at 71 %, followed by the RLC and the MAC sublayers at 23 % and 6 %, respectively. The high processing demand in the PDCP sublayer is mainly caused by the ROHC decompression (71 %) and by the header processing (29 %) measured in total for all PDCP PDUs contained in one transport block.

Correspondent measurements are carried out in the deciphering enabled scenario (c.f. Fig. 11 and Fig. 12). The average L2 DL execution time per transport block is higher compared to the previously presented measurements because of the additional processing demand for the deciphering in the protocol stack model (660 μ s for 32 kB cache). Moreover the dependency of the PDCP sublayer execution time on the total L2 DL execution time is increased to 87 %. The deciphering consumes thereby 68 % of the average PDCP sublayer execution time per transport block followed by the ROHC decompression and the header processing at 23 % and 9 %, respectively. Although the deciphering is carried out in

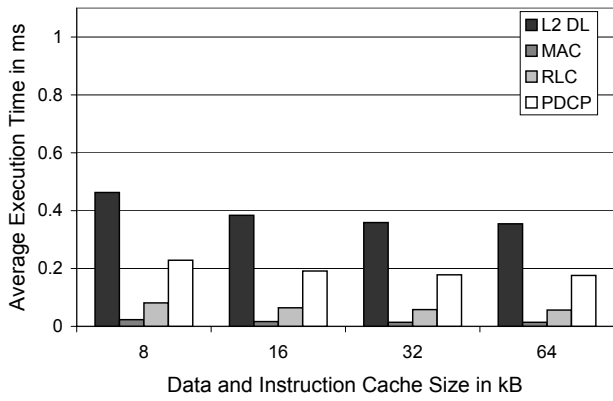


Fig. 9. Average Execution Times of the LTE L2 DL Protocol Stack Model (Deciphering disabled)

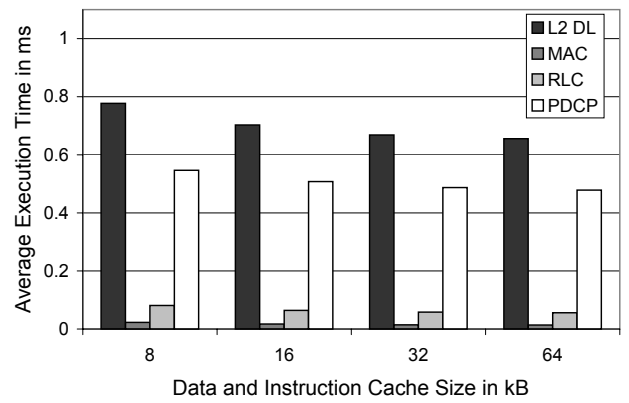


Fig. 11. Average Execution Times of the LTE L2 DL Protocol Stack Model (Deciphering enabled)

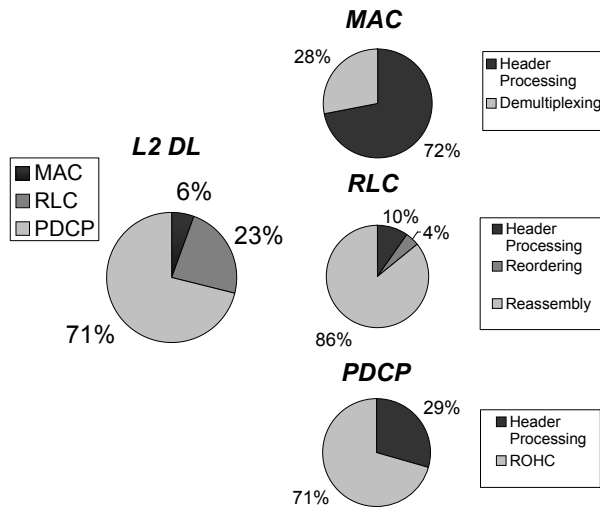


Fig. 10. Relative Distribution of Execution Time for the whole L2 DL (left) and for the sublayers in detail (right) (Deciphering disabled, 32 kB Data and Instruction Cache)

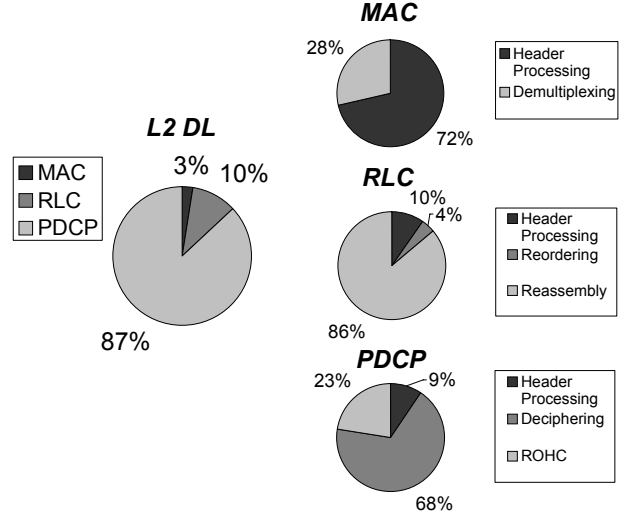


Fig. 12. Relative Distribution of Execution Time for the whole L2 DL (left) and for the sublayers in detail (right) (Deciphering enabled, 32 kB Data and Instruction Cache)

the hardware accelerator, bus demanding data transfers are required for the copying of the data to and from the peripheral, strongly affecting the execution time. Hence, conventional hardware acceleration by only offloading critical algorithms to peripherals, presented in this work exemplarily with the EEA2 hardware accelerator, is an inappropriate technique to meet the requirements of LTE and future mobile devices.

Finally, the impact of the transport block size and the IP packet size on the execution times in the protocol stack model is investigated. First, measurements are carried out by sweeping the transport block size between 40 kbits and 320 kbits (c.f. Fig. 13). With a TTI of 1 ms, the resulting data rates match 40 Mbit/s and 320 Mbit/s, respectively. The computational effort in the protocol stack model is significantly increasing with the transport block size, mainly caused by the algorithms in the PDCP sublayer. Larger transport blocks contain more PDCP PDUs which need to be processed in the correspondent subcomponent of the protocol stack. The parameter sweep over the IP packet size (starting at 250 bytes and ending at

2000 bytes) is depicted in Fig. 14. A fixed transport block size of 100 kbits is thereby configured in the eNodeB/L1 peripheral. The processing effort of the protocol stack model decreases with increased IP packet size. Again, the main cause for the total execution time of the L2 DL is the computational effort in the PDCP sublayer. At a fixed transport block size, small IP packet sizes lead to more PDCP PDUs that are received and processed per transport block.

VI. CONCLUSION

In this paper we present a detailed profiling of the LTE L2 DL protocol processing on an ARM based mobile platform. These obtained results will allow for an efficient hardware/software partitioning. We identify critical algorithms within the protocol stack and analyse the applicability of a conventional hardware acceleration approach for LTE and future mobile terminals. Virtual system prototyping is applied for simulation and analysis of a state-of-the-art mobile hardware platform based on an ARM1176 embedded processor.

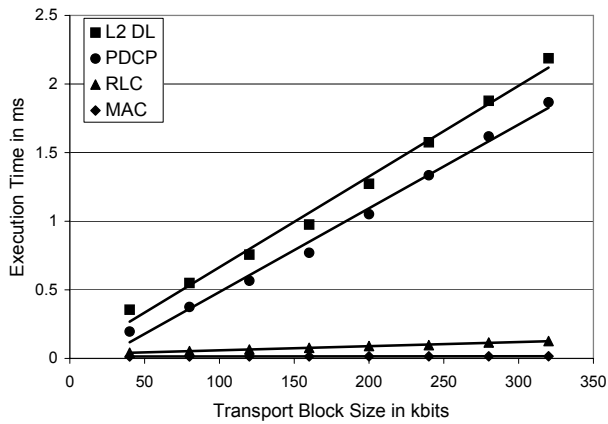


Fig. 13. Average Execution Times of the Protocol Stack Model vs. Transport Block Size (Deciphering enabled, 32 kB Data and Instruction Cache)

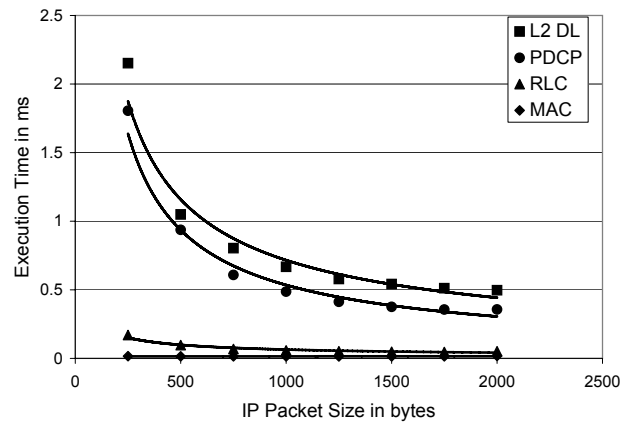


Fig. 14. Average Execution Times of the Protocol Stack Model vs. IP Packet Size (Deciphering enabled, 32 kB Data and Instruction Cache)

Therefore, a configurable base station and physical layer emulator is developed and integrated in the system. By generating LTE transport blocks at different transmission conditions configured by parameters, it allows for a performance analysis of protocol processing on a mobile hardware platform in a very early specification and development phase of wireless communication systems. A protocol stack model representing the most complex and therefore execution time critical part of the LTE L2 DL is implemented and executed in the hardware architecture. The implementation is compliant to the 3GPP Rel.8 specification, whereas the encryption algorithms are accelerated by an EEA2 hardware peripheral. In several simulation scenarios, LTE data rates of 100 Mbit/s and beyond are processed in the protocol stack, in order to obtain a detailed analysis of the execution times. In a pure software mode, with deciphering disabled in the protocol stack model, the average execution time per transport block of the entire L2 DL processing requires more than two-thirds of the available time derived from the worst case transmission conditions in LTE. This high processing demand is mainly caused (71%) by the functionality located in the PDCP sublayer where we identify the ROHC decompression as the major time critical algorithm consuming approximately half of the entire L2 DL execution time. Considering additional computational demand in a complete protocol stack for the L2 uplink and higher protocol stack layers, it becomes clear that a single ARM1176 processor even at high clock frequencies is inappropriate to cope with the high real time requirements. Furthermore, with deciphering activated in the protocol stack we analyse the dependencies of transport block and IP packet sizes on the execution times, whereas the entire L2 execution time is heavily increasing with the number of IP packets located in one transport block. Finally, we show that instead of traditional hardware acceleration concepts more sophisticated hardware accelerators for the L2 are needed to supply enough computational power required in LTE and next generation mobile devices.

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] 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, June 2003
- [2] Berkman, J. Carbonelli, C. Dietrich, F. Drewes, and C. 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, August 2008
- [3] 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, Nov. 2006.
- [4] 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, Apr. 2007.
- [5] *VaST Systems Technology Corporation*, <http://www.vastsystems.com>
- [6] *ARM1176JZF-S Processor Technical Reference Manual*, ARM Limited, Lit.-Nr.: ARM DDI 0301F, 2008
- [7] *AMBA AXI Protocol*, ARM Limited, Lit.-Nr.: ARM IHI 0022B, 2004, <http://www.arm.com>
- [8] *3GPP System Architecture Evolution (SAE): Security Architecture*, 3GPP Std. TS 33.401, Rev. 8.2.0, Dec. 2008.
- [9] 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.
- [10] *The 3rd Generation Partnership Project (3GPP)*, <http://www.3gpp.org>
- [11] *Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification*, 3GPP Std. TS 36.321, Rev. V8.3.0, Sep. 2009
- [12] *Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification*, 3GPP Std. TS 36.322, Rev. V8.3.0, Sep. 2008
- [13] *Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification*, 3GPP Std. TS 36.323, Rev. V8.3.0, Sep. 2008
- [14] *A free implementation of ROHC*, <http://rohc.sourceforge.net>
- [15] *FreeRTOS™*, <http://www.freertos.org/>

- [16] 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, USA, June 2008.
- [17] S. Hessel, D. Szczesny, S. Traboulsi, J. Hausner and A. Bilgic, "On the Design of a Suitable Hardware Platform for Protocol Stack Processing in LTE Terminals", in *7th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC-09)*, Vancouver, Canada, August 2009.
- [18] 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.
- [19] V. Parthasarathy, A.V. Bharathi, V. Rhymend Uthariaraj, "Performance analysis of embedded media applications in newer ARM architectures", in *International Conference Workshops on Parallel Processing 2005 (ICPP 2005)*, Oslo, Norway, pp. 210-214, June 2005.