

# On the Design of a Suitable Hardware Platform for Protocol Stack Processing in LTE Terminals

Sebastian Hessel, David Szczesny, Shadi Traboulsi, Attila Bilgic

Institute for Integrated Systems, Ruhr-Universität Bochum  
D-44780 Bochum, Germany

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

Josef Hausner

Infineon Technologies AG  
D-85579 Neubiberg, Germany

Email: hausner@ieee.org

**Abstract**—In this paper we present a design methodology for the identification and development of a suitable hardware platform (including dedicated hardware accelerators) for the data plane processing of the LTE protocol stack layer 2 (L2) in downlink direction. For this purpose, a hybrid design approach is adopted allowing first investigations of future mobile phone platforms on the system level (using virtual prototyping) combined with more accurate power-area explorations of hardware accelerators on the architectural level. Additionally, we show the employment of an LTE data generator peripheral, realizing L2 uplink processing and thus enabling platform analyses in a closed virtual environment. Furthermore, a modeling technique for a fast and efficient design of virtual hardware accelerator peripherals is demonstrated. A reasonable hardware/software partitioning can thereby be achieved early in the design phase. Once the system architecture is settled and thus the solution space is reduced, VHDL models of the accelerators are developed in order to find a suitable hardware implementation for LTE terminals based on timing constraints by system level simulations. As a case study, the LTE ciphering scheme, including the Advanced Encryption Standard (AES), is applied. We show results of our methodology by developing a deciphering hardware accelerator that enables the LTE protocol stack to process data rates of 100 Mbit/s and beyond.

## I. INTRODUCTION

In modern communication systems, especially in the wireless domain, the continuously growing complexity in terms of new features and services leads to high demands on the underlying hardware. To cope with data rates of up to 100 Mbit/s for the 3G successor Long Term Evolution (LTE) or even beyond (LTE-Advanced), a high computational power is mandatory for the physical base band processing as well as for the data plane of higher layers in upcoming systems. This can be achieved by adopting multi-core processors and/or developing special hardware components for dedicated tasks. Especially for a mobile device with its limited resources like power and on-chip memory, a reasonable hardware/software partitioning is required early in the design phase to meet both, system constraints and economical aspects like time-to-market. For this purpose, hardware/software co-design based on virtual system prototyping is a well-known technique [1] that allows for the exploration of system architectures before the final silicon is available. With the emergence of state-of-the-art processor and peripheral models along with cycle-accurate simulators (provided by tool vendors), virtual system prototyping is established in the embedded system design, even

in industry [2]. It also makes a faster hardware/software co-design possible than real hardware prototyping with Field Programmable Gate Arrays (FPGAs) that is based on low-level descriptions on register transfer (RT) level using hardware description languages (HDLs) like Verilog and VHDL [3]. Once the hardware/software partitioning is done, HDL models are used for the final chip design.

Some research has already been done targeting the base band implementation of LTE terminals [4]. In this paper, however, the identification and design of a suitable hardware platform for the LTE protocol stack layer 2 (L2) is addressed. We therefore propose a design methodology that allows for first investigations of an LTE based mobile phone platform at an early stage of the LTE protocol stack specifications [5]. Our design flow combines two steps: the generation and analysis of a virtual system prototype (VSP) on the system level and the development of power efficient hardware accelerators on the architectural level. With this hybrid approach, the advantages of both levels of abstraction can be employed: a performance optimization of the complete system architecture as well as an accurate power-area exploration of dedicated hardware architectures. The VSP thereby consists of a C model of the protocol stack running on top of a virtual hardware platform. In the protocol stack model, the most time-critical part for the execution of the LTE L2 data plane in downlink (DL) direction is implemented, whereas the core components of the hardware architecture are representative for state-of-the-art mobile phone platforms based on an ARM11 processor [6]. In addition, an LTE data generator peripheral (containing the LTE L2 uplink processing) is included in the VSP to enable a profiling of the protocol stack in a closed environment. Furthermore, the generated data rates can be set above 100 Mbit/s to account for beyond LTE investigations. We show how time-critical functions can be efficiently adopted as hardware peripherals to accelerate software execution. Once a reasonable system setup is found, a further refinement of these peripherals to RT level enables the identification of power and area efficient hardware architectures (based on VSP timing constraints) with higher accuracy and a reduced solution space. Although methods and tools for such estimations on the system level already exist [7]–[9], we use VHDL models to verify and analyze our accelerators, since more accurate results are obtained from the RT level.

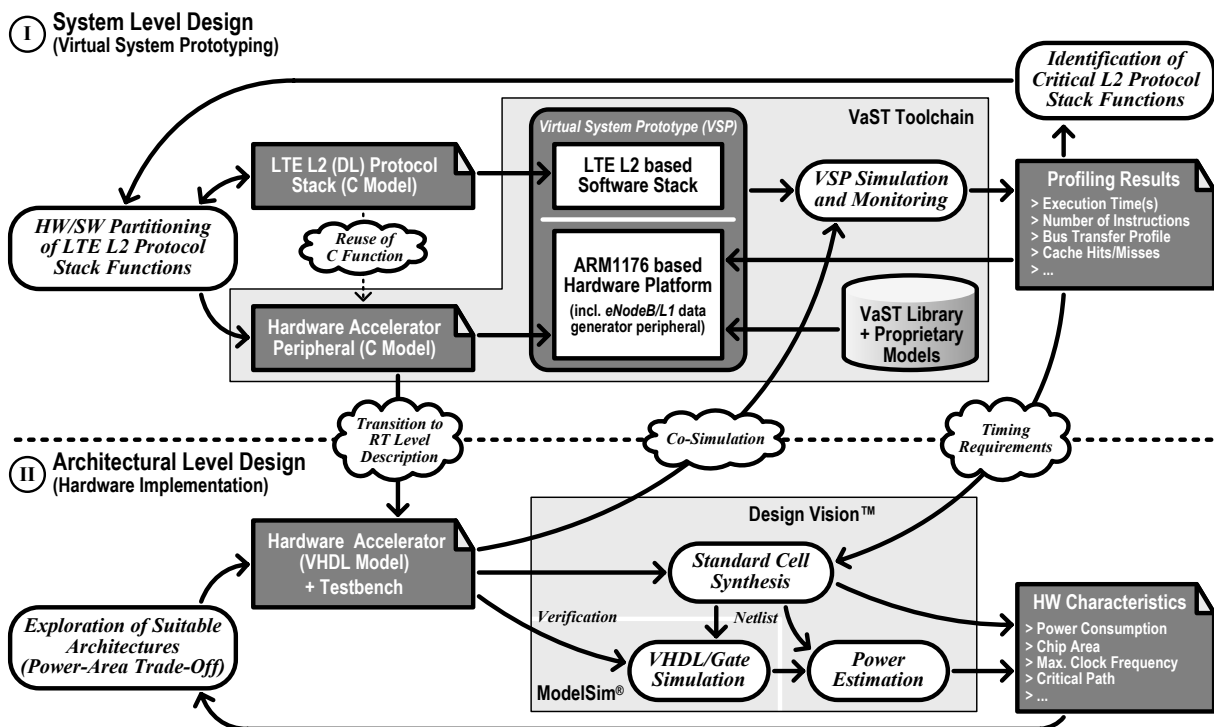


Fig. 1. Design flow for the identification and development of a suitable hardware platform for the protocol layer 2 in LTE terminals. This approach comprises virtual system prototyping of a mobile phone platform (I) as well as the implementation of hardware accelerators based on VHDL models (II).

The paper is organized as follows: Section II presents the design flow, explaining details of the virtual system prototyping and the hardware implementation steps. The VSP of the mobile phone platform is described in section III. Then, a case study regarding the LTE ciphering procedure based on the Advanced Encryption Standard (AES) is provided in section IV, leading to the conclusion in section V.

## II. THE DESIGN METHODOLOGY

Fig. 1 shows the proposed design methodology for the identification and development of a suitable hardware platform for the LTE L2 protocol stack. The two major steps of our hybrid approach, the virtual system prototyping on the system level and the hardware implementation on the architectural level, are described in the following subsections.

### A. System Level Design (Virtual System Prototyping)

For the virtual system prototyping we apply tools from VaST Systems Technology Corporation [10]. Within CoMET<sup>®</sup> an embedded system, which consists of a software stack running on top of emulated hardware, can be simulated. The hardware platform is generated by taking processor and peripheral models from a library provided by the tool vendor. Moreover it is possible to develop proprietary hardware models written in C/C++ or SystemC that can be attached to the bus system of the processor. Software projects can also be built within the tool or suitable binaries are directly

assigned to the hardware platform. The simulation of the VSP in CoMET<sup>®</sup> is cycle accurate. Parameters like cache sizes or latencies can easily be modified between simulation runs. Applying Metrix<sup>™</sup>, timers can be triggered in order to measure the execution time and the number of instructions and clock cycles of dedicated functions in the software stack. Timer tags are therefore placed around these functions in the VSP. Additionally, filters can be used for monitoring bus and memory activities.

Our VSP, described in section III, allows for fast simulations of the mobile phone platform leading to first profiling results of the LTE L2 based software stack (written in C). According to these results, adaptations within the VSP are made to find a reasonable hardware setup that fulfills the required data rates and latencies. This can be done by changing the parameters of the processor platform (like cache sizes or clock frequencies) and/or by accelerating critical protocol stack functions (with respect to their execution time). In the latter case, the identified functions are implemented as peripherals in the hardware platform. The original C models are reused in the design of the hardware accelerator peripherals (see Fig. 2) to enable an efficient hardware/software partitioning. The function body is thereby adopted without adaptations, whereas the arguments in the parameter list of the C function as well as the return value are realized as registers that can be accessed by the software via a VaST bus interface. To achieve high simulation speeds, the processing time is scheduled by a callback function that

is triggered by an activation register. The execution of the function itself, however, is done within one timestamp of the simulated time (see Fig. 2). The processing time can thereby be determined by a parameter in the hardware peripheral, indicating e.g. an absolute time value or the number of clock cycles needed for data processing. Finally, an interrupt is triggered signaling the completion of the function. This mechanism also has the advantage of decoupling the processing time of a function from the simulation time. An example of a hardware accelerator peripheral is given in section III-A. Once a suitable hardware platform is found, timing requirements for the exploration of the hardware accelerators on the architectural level can be obtained.

### B. Architectural Level Design (Hardware Implementation)

To account for a low-power design, energy optimizations should be made on the algorithmic or the architectural level that have the highest potential for power optimizations [11]. In our approach, power and area estimations are carried out on architectural level, since the algorithms for the LTE protocol stack are already fixed by the 3GPP specifications. A straightforward design flow for integrated digital circuits [12] is thereby adopted during the hardware implementation phase. VHDL code is modeled from the selected stack function on RT level. This model is verified by simulations with ModelSim<sup>®</sup> from Mentor Graphics<sup>®</sup>. The synthesis is done with Synopsys' DesignVision<sup>™</sup> Version B-2008.09 based on a standard cell library of Faraday's 90 nm CMOS technology. Hardware characteristics like critical path and chip area can be estimated as well as power values by applying switching activities recorded during gate level simulations with the synthesized netlist. The operating frequency of the hardware module is thereby bounded by the maximum clock frequency (limited by the critical path) and the VSP timing budget. Finally, the netlist can be further processed for the final chip layout. These additional steps (like place and route) are not

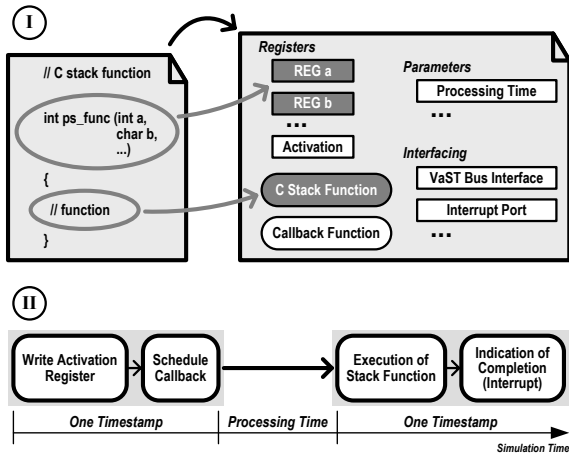


Fig. 2. Modeling of a virtual hardware peripheral by reusing the (C based) protocol stack function (I). The function itself is scheduled by a callback function and executed within one timestamp of simulated time (II).

depicted in Fig. 1. It should be mentioned that a standard wire load model, given by the technology library, is taken for power analysis. To further increase the accuracy, the wire loads from the final layout should be backannotated for the power calculations. In our case, however, the pre-layout estimations are sufficient for the comparison and identification of suitable hardware accelerator architectures, since backannotation is only reasonable once the layout of the complete chip is done.

## III. VIRTUAL SYSTEM PLATFORM

The VSP of the LTE mobile phone platform is the basis for the system level investigations. A simplified block diagram of the VSP is shown in Fig. 3. In the platform, open source models as well as models taken from the VaST model library are adopted. However, proprietary peripherals are also implemented using the C programming language and the Architectural Modeling Programming Interface (AMPI), a C library for hardware modeling provided by VaST. The hardware platform and the software stack are described in the following subsections.

### A. Hardware Platform

The core of the hardware platform is an ARM1176 processor, representative for state-of-the-art processors in mobile phones [13]. It provides an AMBA AXI interface with a 64-bit data and instruction bus, respectively, for accesses to internal and external memories and a 32-bit peripheral bus. The internal memory is used for the initialization of the platform. The external memory, however, is involved during software execution. Read and write latencies are set according to latest mobile phone platforms [6]. Furthermore, a timer model generates operating system interrupts, whereas all interrupts are handled by the interrupt controller. User interaction is provided by a console via a Universal Asynchronous Receiver Transmitter (UART) interface and an emulated display connected to an LCD controller. Additionally, a signal bridge is used to connect VHDL models to the virtual hardware platform.

The eNodeB/L1 data generator [14] represents the testbench for the mobile phone platform to be investigated. Located as a peripheral within the VSP, it simplifies the interaction between stimuli generation and the system under test (see Fig. 4). The eNodeB/L1 model is mainly a configurable emulator for the physical layer signaling and the uplink (UL) counterpart of the LTE L2 protocol stack model (see section III-B). It generates transport blocks from a video file that are accessible by the software stack from internal registers. Settings for the transmission conditions like the data rate or packet loss are user-defined and can therefore be configured for investigations even beyond LTE. Additionally, functions in protocol stack processing can be disabled in order to explore the influence of specific functions on the system performance.

The 128-EEA2 hardware accelerator peripheral is used for stand-alone deciphering with the 128-EEA2 algorithm based on the AES (see section IV-A). In addition to the registers for the initial decryption values (key and state), an internal buffer is used to store the ciphertext (see Fig. 5). The length

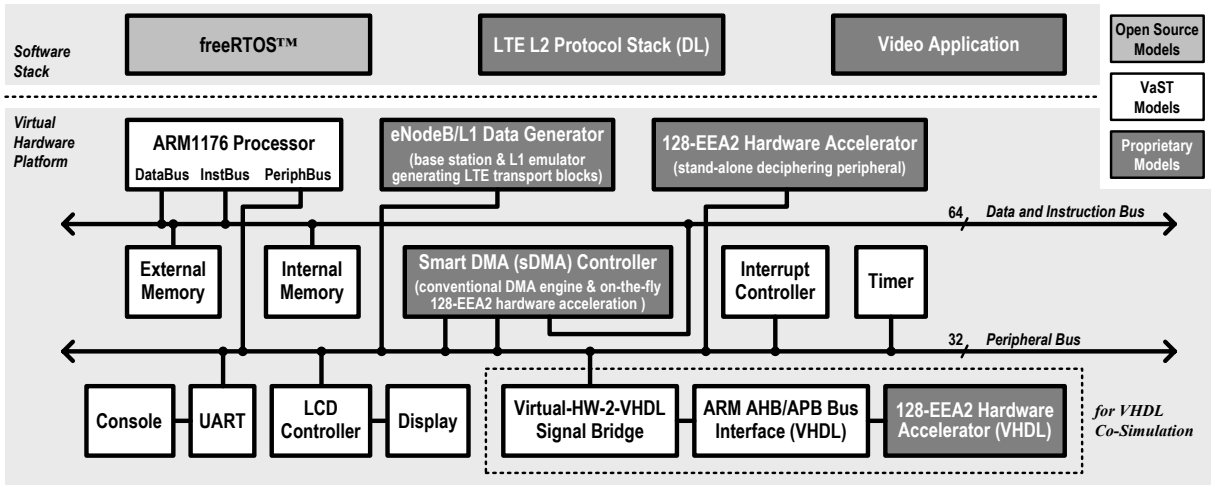


Fig. 3. Virtual system prototype of the LTE mobile phone platform. It consists of a software stack (including an LTE L2 protocol stack model) that runs on top of virtual hardware. Additionally, a signal bridge is used for VHDL co-simulation. Further connecting blocks (like arbiters and converters) are not depicted for a more compact illustration.

(in bytes) of the ciphertext is set by the software. The registers can be accessed from the software via the VaST bus interface which is implemented using the AMPI library for module interaction within the VSP. An interrupt, indicating the completion of the deciphering, is activated after the processing time which is again determined by the deciphering time per byte (of the decryption function) multiplied by the length of the data to be decoded. This concept allows for timing investigations of the hardware accelerator that are independent of the underlying algorithm. Nevertheless, it should be clear that the deciphering algorithm has to match its ciphering counterpart in the eNodeB/L1 peripheral. The smart Direct Memory Access (sDMA) controller peripheral comprises a conventional DMA

engine combined with the 128-EEA2 algorithm to enable on-the-fly hardware acceleration for deciphering [15]. In case of stand-alone deciphering with the 128-EEA2 hardware accelerator, only the DMA engine is used for data copying. Both scenarios are applied for the case study in section IV to show how optimizations of the system architecture are identified with the proposed design methodology.

### B. Software Stack

The software stack contains the most execution-time-critical part of the LTE L2 protocol stack for the data plane in downlink (DL) direction [14], compliant with the 3GPP Rel.8 specifications. The LTE L2 model comprises the Medium Access Control (MAC), the Radio Link Control (RLC) and the Packet Data Convergence Protocol (PDCP) sublayers. The PDCP sublayer is responsible for the decryption of the received data [16]. In our protocol stack model the hardware acceleration for deciphering can be switched on and off in order to compare a pure software implementation with

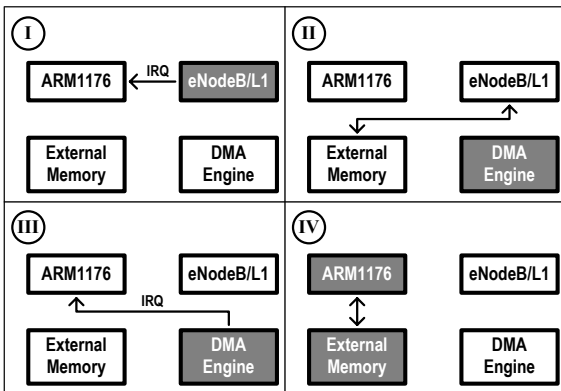


Fig. 4. Simplified simulation flow: A transport block is generated by the eNodeB/L1 and an interrupt is signaled after completion (I). The DMA engine inside the sDMA is invoked that copies the data from the eNodeB/L1 to the external memory (II). After completion, an interrupt is signaled to the ARM1176 processor (III). The ARM1176 starts the protocol stack processing from the external memory (IV). A new transport block is generated after 1 ms.

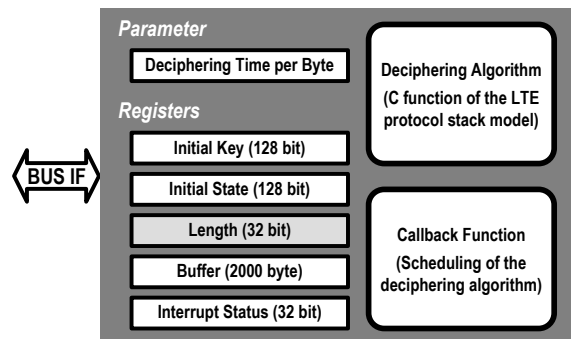


Fig. 5. Structure of the virtual deciphering peripheral (showing registers and parameters). The callback function is activated by writing the *Length* register.

the hardware supported system. When stand-alone 128-EEA2 hardware acceleration is enabled, the DMA engine is invoked to copy the data from the external memory to the peripheral. After deciphering, the DMA engine copies the decrypted data back to the external memory. In case of on-the-fly deciphering, however, the data decryption is done during the first copy operation from the eNodeB/L1 peripheral to the external memory. The initialization of the 128-EEA2 algorithm in both cases is realized in the protocol stack by setting up the initial contents of the decryption registers (see Fig. 5).

The protocol stack consists of several tasks with different priorities, implemented on top of freeRTOS™, an open source real-time operating system. Additionally, a video application decodes the video frames for displaying.

#### IV. THE CIPHERING CASE STUDY

In communication systems, encryption of user data is mandatory for the protection of private contents and for secure accesses to services. The algorithms applied in such systems have a high computational complexity. For this reason they are often realized in hardware, especially in the presence of high data rates. As an example, ciphering in 3G systems (located in the MAC layer) is accelerated by hardware.

In the following subsections, we first introduce the LTE ciphering scheme that is used in this case study, before results from the proposed design flow are presented.

##### A. The AES based Encryption Scheme

The 3rd Generation Partnership Project (3GPP), which is driving the LTE specifications, defines two EPS (Evolved Packet System) Encryption Algorithms (EEA) for user data confidentiality based on a 128-bit input key: SNOW 3G (128-EEA1) and AES (128-EEA2) [17]. While the stream cipher SNOW 3G is already adopted in the 3G confidentiality algorithm f8, the Advanced Encryption Standard (AES) now replaces the KASUMI block cipher. Here, the AES based ciphering scheme is selected for the case study.

The AES algorithm, specified by the National Institute of Standards and Technology (NIST) as a successor of the Data Encryption Standard (DES) in 2001 [18], is a symmetric block cipher based on the Rijndael algorithm. It processes data blocks of 128 bits with a cipher key with lengths of 128, 192 or 256 bits. In LTE, a key length of 128 bits is specified. The structure of the AES algorithm is shown in Fig. 6. It consists of two 128-bit registers for the data (STATE) and the cipher key (KEY) organized in 4x4 byte arrays. Round transformations with the functions *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey* are applied to the data [18]. While *SubBytes* is a nonlinear substitution using the Rijndael S-Box, the *ShiftRows* transformation cyclically shifts the rows of the STATE register with increased offsets. Since the shift operations can be done before the S-Box substitution, the *ShiftRows* function is realized directly in the read and write accesses of the STATE register. With the *MixColumns* transformation the columns of the STATE register are modulo multiplied in Rijndael's Galois field by a given matrix. Finally,

the round key which is generated by the *KeyExpansion* block is added to the data by an XOR operation in the *AddRoundKey* function. The expansion of the key is once again based on the Rijndael S-Box substitution. In order to get the final data,  $N_r = 11$  round transformations are performed while in the first round only the *AddRoundKey* operation is carried out (with the initial data and the initial key) and in the last round the *MixColumns* function is not included.

Several modes of operation for the AES algorithm are recommended by NIST [19]. For LTE, the 3GPP defines the Counter (CTR) mode for user data confidentiality [17]. In the CTR mode (see Fig. 7) the initial data for the AES round transformations is given by a counter block  $T_k$  (with  $k = 1, 2, \dots, i$ ) where  $T_k$  is calculated from counter block  $T_{k-1}$  applying the standard incrementing function defined in [19]. In the initial counter block  $T_1$  the most significant bits are composed of the LTE parameters COUNT, BEARER and DIRECTION [16]. All other bits are set to zero. Based on the cipher key (KEY), a keystream block is received from the final values in the STATE register that is used for the ciphering of the plaintext by an XOR operation. As long as the receiver uses the same input data for the AES algorithm, the plaintext is recovered

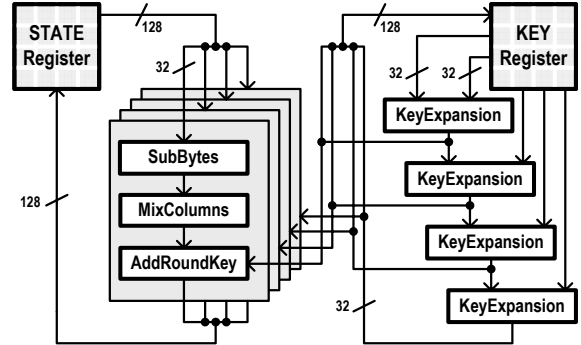


Fig. 6. Structure of the AES algorithm. The data path can be realized with a width of 32, 64 or 128 bits. The *ShiftRows* function is directly implemented in the STATE register accesses and is therefore not depicted.

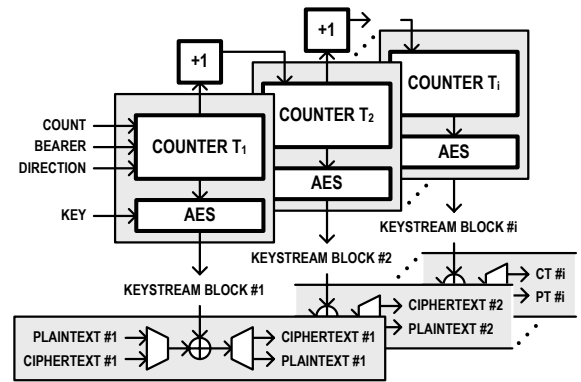


Fig. 7. Structure of the 128-EEA2 ciphering algorithm with the Counter (CTR) mode of operation. The generated keystream block can be used for ciphering of the plaintext as well as for the decryption of the ciphertext.

by deciphering the ciphertext with the same keystream.

Regarding the hardware implementation of the AES algorithm, a good overview can be found in [20]. A straightforward way for variations on architectural level is the adoption of different data path widths [21]. Intuitively a 32-bit data path can be chosen for the AES algorithm, since the *MixColumns* function processes one STATE column of 32 bits at a time. With this approach one round transformation is calculated in four clock cycles resulting in an overall processing time of  $4 \cdot N_r = 44$  clock cycles. For faster implementations, the data path can be extended to 64 bits or 128 bits by increasing the hardware effort (see Fig. 6). The processing time is thereby shortened to 22 and 11 clock cycles, respectively. The identification of a suitable data path width is the primary objective of the architectural analysis in our case study.

### B. Results of the Case Study

1) *System Level Analysis:* As a first step, a software implementation of the 128-EEA2 deciphering algorithm is investigated in the VSP by changing parameters of the hardware platform. During simulations, transport blocks of 100 kbits are generated by the eNodeB/L1 peripheral within the transmission time interval (TTI) of 1 ms to account for an LTE DL data rate of 100 Mbit/s. Fig. 8 shows the execution time (per transport block) of the LTE L2 protocol stack model for different CPU clock frequencies of the ARM1176 processor. The bus clock frequencies are always selected to be 44.4% of the CPU clock frequency corresponding to latest mobile phone platforms [6]. In the diagram, the execution time is halved by doubling the CPU clock frequency. However, it is clear that the processing time limit of 1 ms cannot be reached in this frequency range. For further investigations, the hardware platform is configured with a CPU clock frequency of 450 MHz and a bus clock frequency of 200 MHz in order to have a reasonable trade-off between the processor speed and its power consumption [22].

In another profiling, different cache sizes of the ARM1176 are explored (see Fig. 9). The best performance (with respect to the cache size) is achieved with caches of 32 kB for data and instructions, respectively, as a further increase to 64 kB yields no performance improvement. With this memory setup, the execution time of the protocol stack is reduced to about 8 ms per transport block. Although the C model of the deciphering module is not optimized for software implementation on the ARM1176 processor, the processing time limit of 1 ms is nevertheless exceeded by far. As a consequence, a further profiling is done with a fixed cache size of 32 kB using the decryption peripherals. In Fig. 10 the execution time of the software stack is plotted against the deciphering time per byte for the two different hardware accelerator scenarios. In addition, the eNodeB/L1 peripheral allows for the investigation of higher data rates than specified for LTE. The timing requirement at a data rate of 300 Mbit/s is therefore depicted exemplarily to show a beyond LTE analysis. As expected, the computational demand on the decryption unit is much higher with increased data rates, since the deciphering has to be

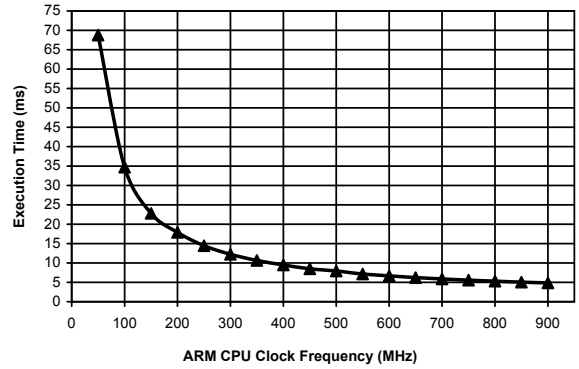


Fig. 8. VSP simulation result showing the execution time of the LTE L2 protocol stack (versus different ARM CPU clock frequencies and a cache size of 32 kB) with the 128-EEA2 deciphering algorithm implemented in software.

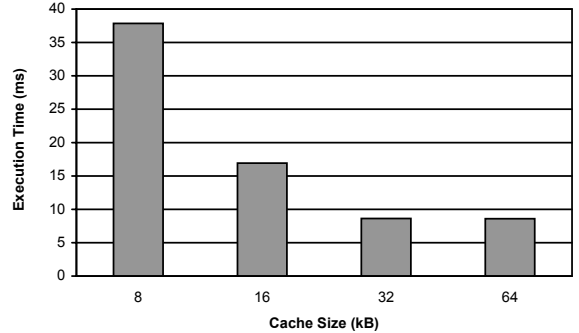


Fig. 9. VSP simulation result showing the execution time of the LTE L2 protocol stack (for different cache sizes) with the 128-EEA2 deciphering algorithm implemented in software.

done much faster to achieve the execution time requirements. Nevertheless, the processing time limit is fulfilled for each system setup at a certain deciphering time per byte, whereas the timing requirements are clearly relaxed using the sDMA controller. This fact is also emphasized looking at the bus activities of both scenarios in Fig. 11. For this analysis, a Metrix™ filter is used in order to trace the clock cycles needed for the bus transfers during L2 protocol stack processing. The arbitration time as well as the transfer time are significantly reduced due to less data copying events adopting on-the-fly deciphering in the sDMA controller. This also indicates the potential power savings, since the communication architecture consumes almost a quarter of the total power consumption in a system-on-chip [23]. So far, a suitable hardware architecture based on the sDMA controller is identified on system level. With the timing constraints shown in Fig. 10, we are able to explore a reasonable architecture for the deciphering function in terms of power consumption and chip area.

2) *Architectural Level Analysis:* Depending on the timing budget from VSP measurements, the absolute power values for the introduced data path widths of the 128-EEA2 decryption block are depicted in Fig. 12. For comparison, the results are once again shown with regard to the deciphering time per byte,

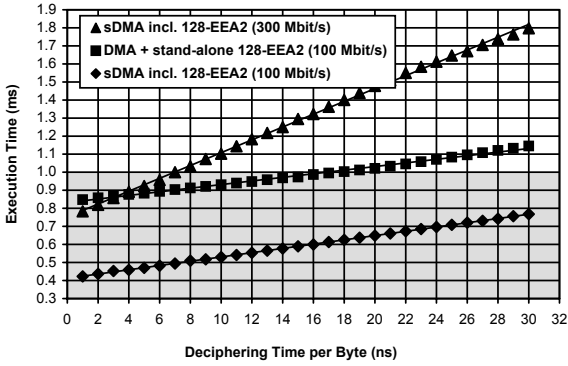


Fig. 10. VSP simulation result showing the deciphering timing constraints using the sDMA scenario (including on-the-fly deciphering) compared to a conventional DMA with a stand-alone deciphering hardware accelerator. The execution time of the software stack must be smaller than 1 ms (grey shaded region).

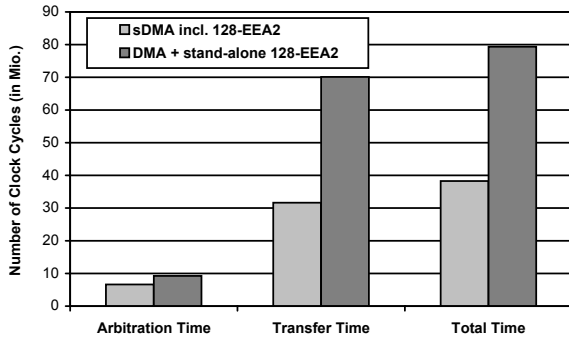


Fig. 11. VSP simulation result showing the number of clock cycles for the overall bus activities comparing the sDMA scenario (including on-the-fly deciphering) with a conventional DMA and stand-alone deciphering.

since the same number of bytes can be processed in less clock cycles using an architecture with a higher data path width. The clock frequencies are adjusted according to the underlying architecture during synthesis. It can be clearly seen that the 128-bit architecture is the best solution from a power-saving perspective. It also enables the lowest deciphering time per byte. On the other hand, the 32-bit data path offers the lowest gate count (see Fig. 13). In the illustrations, the values are given in thousands of gate equivalents (kGE). The number of gate equivalents is calculated from the total area divided by the size of a 2-input AND ( $5 \mu\text{m}^2$ ) that is taken from Faraday's 90 nm CMOS technology library. In order to identify the most suitable architecture with respect to power consumption and area effort, the power-area products of the investigated architectures are shown in Fig. 14. They are plotted for different deciphering times per byte to highlight their dependency on a specific timing requirement (see Fig. 10). Whereas at 24 ns the 32-bit architecture has the lowest power-area product, at 4 ns the 128-bit architecture is the only choice, since the other solutions are not applicable at this data rate. As a result, the latter architecture is the preferable solution for LTE terminals,

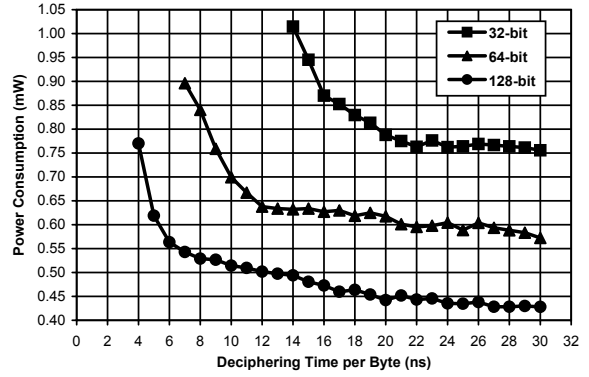


Fig. 12. Power consumption of the 128-EEA2 for different AES architectures (in terms of data path widths) based on VHDL models.

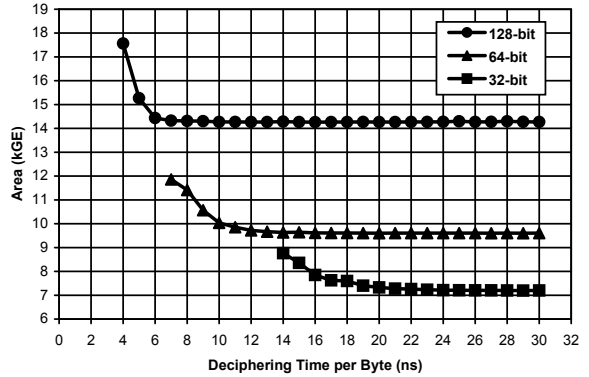


Fig. 13. Area utilization of the 128-EEA2 for different AES architectures (in terms of data path widths) based on VHDL models.

for it offers the lowest power consumption and the highest data throughput. The area overhead is acceptable due to increasing integration capabilities of today's and future chips. For the sake of completeness, the VHDL models are finally verified in the VSP as illustrated in Fig. 3.

## V. CONCLUSION

In this paper we propose a design methodology that allows for a fast analysis and development of a suitable hardware platform for the LTE protocol stack processing of layer 2 (L2) in downlink direction. In our hybrid approach, we combine the system level design with the architectural level design to benefit from both approaches: high level investigations of the complete system architecture as well as accurate explorations of dedicated hardware modules. By virtual prototyping of an LTE based mobile phone platform (with an ARM11 processor), critical protocol stack functions (with respect to their execution time) can be identified for hardware acceleration early in the design phase. This is supported by an LTE data generator peripheral (including L2 uplink processing) that is located in the virtual hardware and therefore allows for realistic performance analyses in a closed environment. Additionally, we show a technique for the design of hardware accelerator

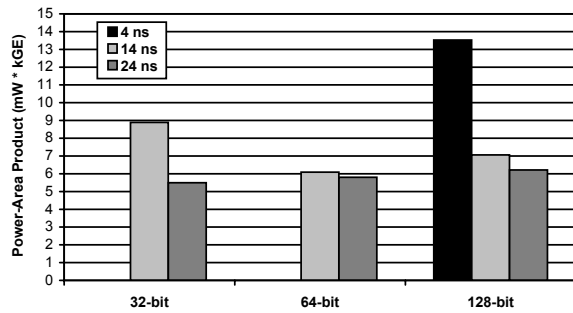


Fig. 14. Power-area product at different timing requirements for the investigated AES architectures (in terms of data path widths) in the 128-EEA2 based on VHDL models. The 32-bit and the 64-bit architecture, respectively, are not applicable at 4 ns and therefore the corresponding black bars are missing.

peripherals with a short development time (since we reuse the original C functions) and high simulation speeds. This results in an efficient hardware/software partitioning which can be optimized and verified in the same virtual environment. Once the solution space is reduced, a further refinement of the accelerator modules to register transfer level (using VHDL models) offers more accuracy when performing a power-area investigation on the architectural level. By this analysis, a suitable hardware implementation for LTE terminals can be found, based on the timing budget of virtual platform simulations. The proposed design methodology is proven showing results from the identification and development of a deciphering hardware accelerator using the Advanced Encryption Standard (AES). With our hybrid approach, other L2 functionalities can be analyzed and realized for hardware acceleration (if applicable), leading to a reasonable overall system architecture for the LTE L2 protocol stack. Furthermore, the design flow in combination with the presented data generator peripheral will be used in the future for further investigations of beyond LTE mobile phones. Additionally, our methodology can be adopted even for the analysis of data processing in other layers of the modem subsystem. In general, the proposed design flow is useful and beneficial for the development of a suitable hardware platform for comparable embedded systems.

#### ACKNOWLEDGMENT

The authors acknowledge the excellent cooperation with all project partners in the EASY-C project and the support by the German Federal Ministry of Science and Education (BMBF).

#### REFERENCES

- [1] 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.
- [2] 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.
- [3] 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 and Test in Europe (DATE 2007)*, Nice, France, pp. 1-3, Apr. 2007.

- [4] 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 (IWCMC 2008)*, Crete Island, Greece, pp. 970-975, Aug. 2008.
- [5] The *3rd Generation Partnership Project (3GPP)* website, available: <http://www.3gpp.org/ftp/Specs/html-info/36-series.htm>
- [6] S. Hessel, F. Bruns, A. Bilgic, A. Lackorzynski, H. Härtig and J. Hausner, "Acceleration of the LA/Fiasco Microkernel Using Scratchpad Memory", in *International Workshop on Virtualization in Mobile Computing (MobiVirt 2008)*, Breckenridge, USA, June 2008.
- [7] L. Benini and G. de Micheli, "System-Level Power Optimization: Techniques and Tools", in *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Volume 5, Issue 2, pp. 115-192, Apr. 2000.
- [8] N. Bansal, K. Lahiri, A. Raghunathan and S. T. Chakradhar, "Power Monitors: A Framework for System-Level Power Estimation Using Heterogeneous Power Models", in *18th International Conference on VLSI Design (VLSID 2005)*, Kolkata, India, pp. 579-585, Jan. 2005.
- [9] A. Stammermann, L. Kruse, W. Nebel and A. Pratsch, "System Level Optimization and Design Space Exploration for Low Power", in *14th International Symposium on Systems Synthesis (ISSS 2001)*, Montréal, Canada, pp. 142-146, 2001.
- [10] The *VaST Systems Technology Corporation* website, available: <http://www.vastsystems.com>
- [11] M. Schämamm, M. Bücken, S. Hessel and U. Langmann, "Low-Power Design on Algorithmic and Architectural Level: A Case Study of an HSDPA Baseband Digital Signal Processing System", in *Design, Automation and Test in Europe (DATE 2007)*, Nice, France, pp. 1406-1411, Apr. 2007.
- [12] D. Jansen et al., "The Electronic Design Automation Handbook", Kluwer Academic Publishers, 2003.
- [13] O. Silven and K. Jyrkkä, "Observations On Power-Efficiency Trends in Mobile Communication Devices" in *EURASIP Journal on Embedded Systems*, Vol. 2007, Article ID 56976, 2007.
- [14] D. Szczesny, A. Showk, S. Hessel, U. Hildebrand, V. Frascaolla and A. Bilgic, "Performance Analysis of LTE Protocol Processing on an ARM based Mobile Platform", submitted in *11th International Symposium on System-on-Chip (SoC 2009)*, Tampere, Finland, Oct. 2009.
- [15] D. Szczesny, S. Hessel and A. Bilgic, "On-the-fly Hardware Acceleration for Protocol Stack Processing in Next Generation Mobile Devices", submitted in *5th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2009)*, Grenoble, France, Oct. 2009.
- [16] *Evolved Universal Terrestrial Radio Access (E-UTRA): Packet Data Convergence Protocol (PDCP) specification*, 3GPP Std. TS 36.323, Rev. 8.4.0, Dec. 2008.
- [17] *3GPP System Architecture Evolution (SAE): Security Architecture*, 3GPP Std. TS 33.401, Rev. 8.2.0, Dec. 2008.
- [18] National Institute of Standards and Technology (NIST), "FIPS Publication 197: Advanced Encryption Standard (AES)", Nov. 2001, available: <http://csrc.nist.gov/publications/PubsFIPS.html>
- [19] National Institute of Standards and Technology (NIST), "NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation - Methods and Techniques", Dec. 2001, available: <http://csrc.nist.gov/publications/PubsSPs.html>
- [20] M. Alam, S. Ghosh, D.R. Chowdhury, I. Sengupta, "Single Chip Encryptor/Decryptor Core Implementation of AES Algorithm", in *21st International Conference on VLSI Design (VLSID 2008)*, Hyderabad, India, pp. 693-698, Jan. 2008.
- [21] A. Satoh, S. Morioka, K. Takano, S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization", in *7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT 2001)*, LNCS Vol. 2248, pp. 239-254, 2001.
- [22] K. Flautner, S. Reinhardt and T. Mudge, "Automatic Performance Setting for Dynamic Voltage Scaling", in *Journal of Wireless Networks*, Volume 8, Issue 5, pp. 507-520, Sep. 2002.
- [23] K. Lahiri and A. Raghunathan, "Power Analysis of System-Level On-Chip Communication Architectures", in *2nd International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2004)*, Stockholm, Sweden, pp. 236-241, Sep. 2004.