Hindawi Publishing Corporation
VLSI Design
Volume 2013, Article ID 369627, 15 pages
http://dx.doi.org/10.1155/2013/369627

*Research Article*

# Energy-Efficient Hardware Architectures for the Packet Data Convergence Protocol in LTE-Advanced Mobile Terminals

**Shadi Traboulsi, Valerio Frascolla, Nils Pohl, Josef Hausner, and Attila Bilgic**

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

Correspondence should be addressed to Shadi Traboulsi; shadi.traboulsi@rub.de

Received 4 September 2012; Accepted 24 November 2012

Academic Editor: Jose Carlos Monteiro



In this paper, we present and compare efficient low-power hardware architectures for accelerating the Packet Data Convergence Protocol (PDCP) in LTE and LTE-Advanced mobile terminals. Specifically, our work proposes the design of two cores: a crypto engine for the Evolved Packet System Encryption Algorithm (128-EEA2) that is based on the AES cipher and a coprocessor for the Least Significant Bit (LSB) encoding mechanism of the Robust Header Compression (ROHC) algorithm. With respect to the former, first we propose a reference architecture, which reflects a basic implementation of the algorithm, then we identify area and power bottle-necks in the design and finally we introduce and compare several architectures targeting the most power-consuming operations. With respect to the LSB coprocessor, we propose a novel implementation based on a one-hot encoding, thereby reducing hardware's logic switching rate. Architectural hardware analysis is performed using Faraday's 90 nm standard-cell library. The obtained results, when compared against the reference architecture, show that these novel architectures achieve significant improvements, namely, 25% in area and 35% in power consumption for the 128-EEA2 crypto-core, and even more important reductions are seen for the LSB coprocessor, that is, 36% in area and 50% in power consumption.

## 1. Introduction

New data-demanding mobile applications, such as video streaming and online gaming, are the main drivers for higher mobile data rates. New standards improve the mobile Internet experience by providing downlink data rates starting from 100 Mbit/s in LTE up to 1 Gbit/s for LTE-Advanced [1]. This huge increase in data rates imposes new challenges on the design of mobile devices, where computational power and battery lifetime are strictly limited.

A recent uplink/downlink performance analysis of an LTE protocol stack on a representative virtual mobile platform [2, 3] has identified the Protocol Data Convergence Protocol (PDCP) as the most time-critical component within the Layer 2 software architecture. PDCP incorporates two computationally expensive tasks: the Robust Header Compression (ROHC) algorithm, which compresses IP packet headers in order to improve the spectral efficiency of radio links, and the ciphering algorithm, responsible for user data protection and for providing a secure communication towards the core network. While both protocol functions show long processing times, ciphering comes in the first place followed by ROHC. Apart from performance requirements, PDCP algorithms must be designed for low-power in order to enable a longer battery lifetime, a crucial feature for mobile devices. As more and more functionalities are being integrated in mobile handsets, area-optimized designs are also required to reduce chip costs.

Two architectural approaches exist to accelerate computationally expensive processing functions. The first one is based on embedded multicore processing to speedup software execution of protocol stack functionalities [4–7]. This approach provides flexibility and reuse of computational resources as processor cores can be shared by several applications, including the protocol stack. However, it might be very challenging to identify and exploit parallelism, as most of the currently used mobile algorithms were not designed with the aim of running in a parallel architecture [8]. The second approach is based on the design of

dedicated hardware accelerators in the form of cores. These hardware blocks are connected to an on-chip interconnect to offload the communication processor and boost processing speed, in order to satisfy the targeted timing requirements.

In addition to the above mentioned issues on parallelism, the software approach requires scaling the number of processor cores in order to increase the processing performance. This does not fit well with the tight area and power constraints of mobile handsets, especially if we consider LTE-Advanced terminals, with data rates in the order of 1 Gbit/s. Another advantage of the hardware approach over its software counterpart is its ability to achieve very low absolute power consumption levels by appropriately tweaking its architecture internals [9, 10]. Consequently, also for the next few years, mobile devices based on heterogeneous multicore platforms will be mainstream [11].

*1.1. Main Contributions.* Depending on the domain taken into consideration, Register Transfer Level (RTL) hardware architectures focus either on high speed cores or on low-power consumption designs. As the latter is a paramount key performance indicator for mobile devices, the focus of this work is on the design of low-power architectures for the PDCP protocol within the Layer 2 of a cellular protocol stack. Our contribution is summarized as follows.

(i) We develop an energy-efficient crypto-core for the 128-EEA2 confidentiality algorithm which is part of the PDCP protocol. This is achieved by optimizing the power of the core cipher on which 128-EEA2 is based, that is, the Advanced Encryption Standard (AES) in counter mode. For that purpose, we introduce several power-efficient architectures and evaluate them in terms of area and power consumption.

(ii) We present a configurable data path architecture for the 128-EEA2 crypto-core to limit the peak power in modern power-constrained mobile devices. The hardware adapts its peak power seamlessly at run time by switching between several architectural modes.

(iii) We propose a novel, compact, and energy-efficient hardware architecture for the Least Significant Bit (LSB) encoding, the core compression mechanism in the ROHC algorithm. The architecture is evaluated against a reference architecture.

*1.2. Paper Organization.* The rest of the paper is organized as follows: in Section 2, we present a brief literature survey. In Section 3, we describe both confidentiality (128-EEA2) and ROHC algorithms. In Section 4, critical functional blocks in 128-EEA2 are identified and various architectural optimizations are presented. Section 5 describes the LSB coprocessor with techniques for power reduction. We evaluate the proposed architectures and discuss the results in Section 6, leading to the conclusion in Section 7.

## 2. Related Work

Ciphering algorithms in mobile handsets are typically accelerated by dedicated hardware [12–15]. Hardware architectures for core ciphers of several LTE confidentiality algorithms were separately implemented in various works [16–18], where the main focus was to deliver high speed FPGA or ASIC designs. Few recent works [19, 20] proposed compact and power-efficient architectures for both 128-EEA1 and 128-EEA3 ciphering algorithms. However, previous works proposing low-power implementations of AES, the core cipher of 128-EEA2, are limited to the byte substitution function [21–24]. To our best knowledge, no research work has ever introduced hardware power reduction techniques for AES especially targeting its mode of operation in the LTE protocol. However, there are some attempts to accelerate both 128-EEA1 and 128-EEA2 algorithms by parallelizing them on an embedded ARM multicore processor [25, 26]. The focus in these works was first to deliver an efficient implementation with the minimum possible instruction count. Parallelism was then investigated and deployed at both algorithmic level (128-EEA1) and block level (128-EEA2) for further optimization of the processing time. In [25], different message blocks are dispatched to available cores for parallel processing. Energy savings were also achieved compared to single core implementations by exploiting the excess in performance gain for frequency and voltage scaling [9], or by mapping protocol stack software to several small/big cores associated with different speeds and supply voltages [27]. Although the above mentioned works do not require any additional resources in a multicore-based handset, they were not able to achieve processing throughputs fulfilling LTE-Advanced requirements.

While some works proposed hardware architectures for ciphering algorithms, very few considered hardware designs for Robust Header Compression (ROHC). Two acceleration concepts for ROHC are presented in [28]: scratch pad memories and dedicated hardware support. These concepts were investigated at high-level with system simulations without suggesting concrete hardware architectures. Another work in [29] focused on implementing a high-performance hardware architecture for an outdated release of ROHC in network processors, where energy consumption is not an issue. In [30], an architecture for ROHC version 2 (ROHCv2) hardware with multiple-flow support was introduced for LTE devices. It investigates the advantages of ROHCv2 hardware by comparing it to its software counterpart on an embedded FPGA-based platform. The proposed hardware architecture is however too large and does not suggest any power reduction for the Window-based LSB encoding function, which consumes two thirds of the total hardware power [30].

The ROHC algorithm contains many memory checking operations; therefore, implementing its functionality in hardware implies a huge design, which requires transfer of all headers through an on-chip bus. Moreover, it is not efficient due to both the considerable power consumption levels of interconnects [31] and the high transfer time penalty caused by bus congestions. In this work, instead, we suggest to partition ROHC into software and hardware

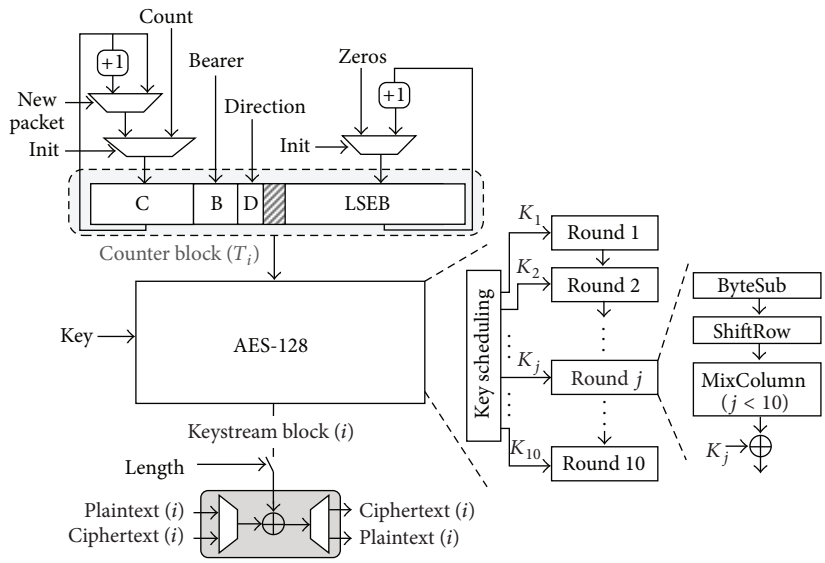


Figure 1: Structure of the 128-bit EEA2 ciphering algorithm.

[32]. The software part executes tasks responsible for memory checking, as packet header fields typically reside in the processor's cache memory, whereas the hardware part realizes the most time and power consuming operation, that is, the least significant bit (LSB) encoding functionality.

## 3. Packet Data Convergence Protocol

The Packet Data Convergence Protocol (PDCP) resides in Layer 2 of the LTE cellular communication protocol stack [33]. It is part of the data plane and receives user data in the form of IP packets during uplink communications. First, the headers of incoming packets are compressed using the specified Robust Header Compression (ROHC) algorithm. After that, IP packets are ciphered using one of the standardized (128-EEA1, 128-EEA2, and 128-EEA3) confidentiality algorithms [34]. A certain encryption algorithm is selected when the connection is established, depending on the control messages signaled by the Radio Resource Control (RRC) in the control plane. A PDCP header is then added to each ciphered packet before being fed to the next downstream protocol, the Radio Link Control (RLC) sublayer. At the receiver side, PDCP performs the reverse functionality represented by deciphering and header decompression operations. In the following, both ciphering and compression algorithms considered in our study are described in more detail.

*3.1. 128-EEA2 Ciphering Algorithm.* The Evolved Packet System Encryption Algorithm (128-EEA2) is one of the three confidentiality algorithms specified in LTE [34]. It is based on the 128-bit Advanced Encryption Standard (AES) running in the counter (CTR) mode [35, 36]. The structure of the 128-EEA2 confidentiality algorithm is shown in Figure 1. The 128-bit counter block ($T_i$) is used as an input to be processed by AES. The content of this block is initialized with communication parameters such as packet count (COUNT), radio bearer (BEARER), and the direction of communication (DIRECTION). The output of AES is then combined with the user data using the XOR operation to generate the so-called ciphertext (uplink encryption) or plaintext (downlink decryption). The PDCP packet length (LENGTH) is additionally used to strip out unwanted bits in case the packet being ciphered is not a multiple of the block size (128 bits). The counter mode is characterized by incrementing the Least Significant Eight Bytes (LSEB) value of $T_i$ by one upon ciphering of each data block within a PDCP packet. In addition, the COUNT value in $T_i$ is incremented by one before processing a new PDCP packet.

The AES cipher operates on the counter block organized in a $4 \times 4$ byte matrix, referred to as *state*. The cipher consists of ten successive rounds preceded by a key addition operation (not shown in figure). Each round is based on four operations: ByteSub, ShiftRow, MixColumn, and AddKey. The last round however performs no MixColumn operation. A key scheduling function is applied on every cipher key (KEY) update to generate the keys ($K_j$) associated with each round. The ByteSub performs a nonlinear byte-to-byte substitution with the Rijndael S-Box. Each row $m$ in a state is then rotated to the left by $m$ bytes in the ShiftRow operation. The MixColumn function follows with modulo multiplication of the *state* columns by rotated versions of the polynomial vector (03 01 01 02). Finally, the AddKey combines the *state* bytes with the corresponding round key using XOR operations. The result of each round is fed to the next round till the keystream block is delivered by the last round.

The cipher Key and all communication parameters are signaled by the Radio Resource Control (RRC) sublayer. The value of KEY is usually assigned at the beginning of a communication session and is unusual to change during

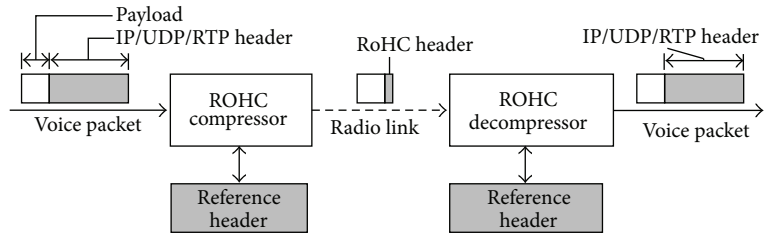


Figure 2: Example demonstrating the concept of ROHC.

transmission. However, it might be reassigned by RRC during cell or base station handover scenarios.

*3.2. Robust Header Compression.* Robust Header Compression (ROHC) provides an efficient use of radio links by exploiting the redundancy in the packet headers over a certain flow. It reduces the packet header sizes, and thus the allocated wireless links bandwidth, before packet transmission. The example in Figure 2 illustrates the potential of ROHC, where a mobile handset transmits voice packets using IPv4/UDP/RTP. High quality voice packets have a typical payload size of 20 bytes and a 40 bytes header. ROHC achieves a header size between one and four bytes, providing an optimum overhead reduction by a factor of 40. As a result, the effective communication bandwidth is increased. The ROHC framework [37] was specified by the Internet Engineering Task Force (IETF) [38]. An optimized version named ROHCv2 [39] was subsequently released with modifications aiming to improve robustness against high packet loss and retransmissions in wireless links. The ROHC framework defines encoding methods for several profiles. For the purpose of our study, we will focus on the most complex IP/UDP/RTP profile.

Each packet header contains many fields, some are fixed (static) and others may change rarely or predictably (dynamic) during a communication session. A reference copy of all header fields is stored at both compressor and decompressor to ensure a lossless transmission. The compressor starts operating in the Initialization and Refresh (IR) state, where a number of complete packet headers are sent for synchronization with the decompressor. Afterwards, a transition to the Compression (CO) state is made and compressed packet headers are sent with Cyclic Redundancy Check (CRC) protection. If no successful decompression is verified by CRC, then more header information will be transmitted to repair the context at the decompressor. In the bidirectional mode, a negatively acknowledged feedback signal (SNACK or NACK) is sent by the decompressor, requesting retransmission of full header context (IR) packets or dynamic context (CO-Repair) packets. However, in Unidirectional mode where a feedback channel is not provided, IR packets are sent periodically to refresh the reference context at the decompressor, resulting in less efficient compression.

Compressed packets are generated by encoding two dynamic fields: the IP Identifier (IP-ID) and the RTP Timestamp. An offset (IP_ID_DELTA) is computed by subtracting the Message Sequence Number (MSN) from IP-ID. Since

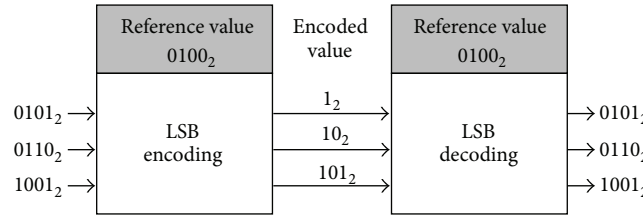


Figure 3: Example illustrating the concept of LSB encoding.

real-time audio and video applications employ codecs with fixed sampling periods, the RTP Timestamp usually increases by a fixed value (TS_STRIDE) and hence can be expressed with a variable number (TS_SCALED) multiplied by the stride. Both TS_SCALED and IP_ID_DELTA are further encoded with the Least Significant Bit (LSB) scheme, where only the difference between the value to be encoded ($v$) and a successfully received previous value ($v_{\text{ref}}$) is sent, as illustrated in Figure 3. LSB defines the following interpretation interval:

$$f(v,k) = \left[v_{\text{ref}} - p, v_{\text{ref}} + \left(2^k - 1\right) - p\right], \tag{1}$$

where $k$ is the number of least significant bits to be sent. The value of $k$ must be calculated so that $v$ lies in the interpretation interval. This function which determines $k$ is referred to as $g(v_{\text{ref}}, v)$. A shifting parameter $p$ might be tuned according to channel conditions to improve compression efficiency. To improve the robustness of LSB encoding, a Window-based LSB (WLSB) is applied by employing a sliding window which stores a certain number of previous reference values. The value of $k$ is then determined using the lower ($v_{\text{ref min}}$) and upper ($v_{\text{ref max}}$) bounds of the sliding window as follows:

$$k = \max\left(g\left(v_{\text{ref min}}, v\right), g\left(v_{\text{ref max}}, v\right)\right). \tag{2}$$

WLSB guarantees correct decoding of $v$ as long as both $v_{\text{ref}}$ at the compressor and decompressor lie in the $[v_{\text{ref min}}, v_{\text{ref max}}]$ interval.

## 4. 128-EEA2 Crypto-Core

The hardware architecture of the EEA2 core is depicted in Figure 4. It is basically composed of three main blocks: control unit, counter block, and an AES engine. The latter consists of a Galois Field (GF(2)) counter required for the computation of the round keys by the key scheduler. In addition, the control unit is adapted in such a way that the

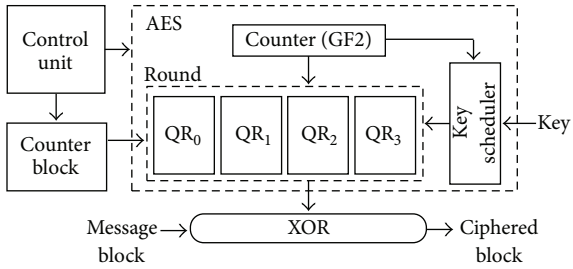


Figure 4: Hardware architecture of the 128-EEA2 core.

same counter is also used to track the number of encryption rounds executed by the round module. Round processing is split into four parallel quarter rounds ($QR_0$ to $QR_3$), each of which processes a 4-byte chunk, that is, a state's column, of the message block. The counter block is a memory-mapped IO 128-bit register. It can be directly accessed by software and updated with a new initialization vector before encryption is triggered. Synchronization between modules is maintained by the control unit, which checks the input message length and activates potential blocks to ensure a complete encryption of the message.

In order to analyze the improvements with respect to area and power consumption of the 128-EEA2 core, a basic hardware architecture was implemented as a reference design. This basic architecture was designed without any special architectural optimizations and realizes the Sboxes as lookup tables (LUT) using several multiplexers. It is described as a VHDL model and synthesized using Faraday's standard cell library of 90 nm with a core power supply of 1 V.

Figure 5 displays the power and area breakdown of the 128-EEA2 core across its processing modules. In this figure, both control and counter block modules are grouped together with state registers under one category (Control + Reg). Other categories represent functionalities in the AES engine. In particular, SubBytes, MixColumn, and AddKey belong to the Round module. A large portion of the power consumption (39%) is consumed by SubBytes as it contains sixteen substitution boxes (Sbox). KeySchedule follows with 31% of core's power which is relatively high taking into consideration that it contains only a quarter of the number of Sboxes used in SubBytes. This can be explained by the fact that logic switching in KeySchedule's Sboxes is rather heavy because of the fact that consecutive round keys are highly uncorrelated, so to follow the need to provide a higher level of security. In addition, most of the core's area is occupied by SubBytes (55%) followed by KeySchedule (21%). As a result, we identify both round and key scheduler as the most power-critical blocks, where Sboxes have a considerable share of power and area in both modules. In the following, we present some optimization techniques focusing on these three components in order to reduce the overall area and power consumption.

*4.1. Substitution Box.* To optimize the Sboxes, we have investigated three architectural alternatives to the LUT reference implementation. The first is based on a Decoder-Switch-Encoder architecture (DSE) [23]. Within this architecture, the transformation value is generated in three steps. First, an 8-bit input word is one-hot encoded using a decoder. The generated 256 bits then enter a switch and are routed to another 256-bit one-hot code according to Rijandel's Sbox input/output mappings. Finally, the switch output is converted from a one-hot encoded scheme back to an 8-bit binary representation, which holds the output transformation value. As a second alternative, we propose a Nibble-Based Decoder-Switch-Encoder (NBDSE) architecture which splits the input byte into high and low nibbles ($N_H$ and $N_L$) to reduce Sbox complexity, as shown in Figure 6. The transformation of each nibble is computed in parallel to obtain the low ($S_L$) and high ($S_H$) nibbles of the substitution value. In this context, decoded high and low nibbles can be perceived as column and row lines respectively. Nibble switches consists of a permutation block which maps decoded row lines ($R_x$) to sixteen sub-switches (highlighted in grey) based on an offline analysis of the Sbox contents. Figure 6 depicts the structure of a sub-switch containing three levels of logic. Row nibble transformations within a column sharing the same value are grouped through ORs followed by AND gates as depicted in Figure 6. The output of the sub-switches forms a 16-bit one-hot code which is fed into a 16-to-4 bit encoder to deliver the nibble's transformation value.

Finally, the third architecture, a Correlation-Based Implementation (CBI) of the Sbox, realizes a logic array of 16 rows by 16 columns. Within a row, each four neighboring transformations are grouped and replaced by the output of a switch ($SW_i$) as shown in Figure 7. A 16-bit pattern word is computed once for all rows in the Sbox from the lower column address bits, whereas the upper bits control the row multiplexer to select a corresponding switch output. To avoid unnecessary switching in the multiplexers of nonselected rows, the pattern bits are masked with decoded column address bits. The wiring of individual switch outputs with the masked pattern bits is obtained through an offline analysis of Sbox contents, where the correlation between grouped transformations is studied. A two-stage output multiplexer uses the 4-bit row address to deliver the final transformation value.

*4.2. Key Caching.* According to the LTE protocol [33, 34], the cipher key usually changes during cell handovers and is rarely altered within a communication session. In other words, many PDCP packets and their corresponding counter blocks will share the same cipher key within a single connection. We exploit this property to reduce the power consumed by caching round keys so that other packets can reuse them during an established connection. Figure 8 depicts the architecture of the key scheduler, where a RAM memory is inserted to cache the computed round keys for reuse in successive counter encryptions. The cache memory is implemented as a register set containing ten 128-bit registers, each dedicated for storing one round key. The control block is extended to read any changes in the cipher key assigned by the radio resource control in the control plane. If a new cipher key ($K_{\text{in}}$) is received, the round key computation mechanism is activated for the first counter block encryption. Simultaneously, the round keys ($K_r$) are cached in their corresponding registers pointed by the address generator. In

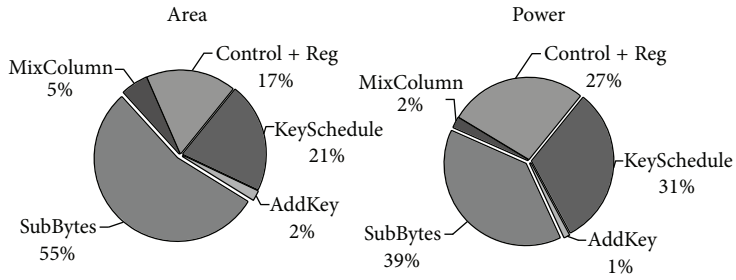


Figure 5: Breakdown of the area and power consumed by the 128-EEA2 core.

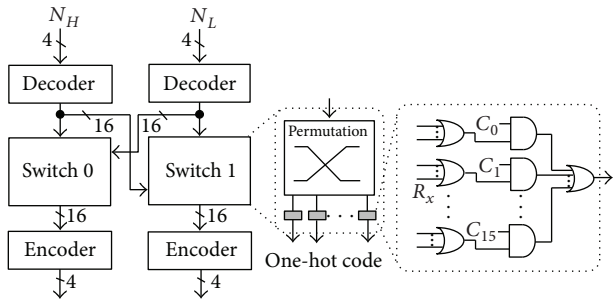


Figure 6: Structure of the NBDSE Sbox architecture.

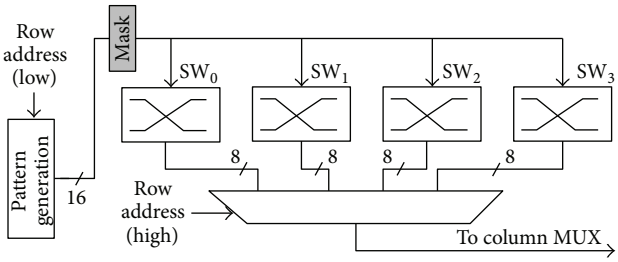


Figure 7: Structure of the rows in the Sbox CBI architecture.

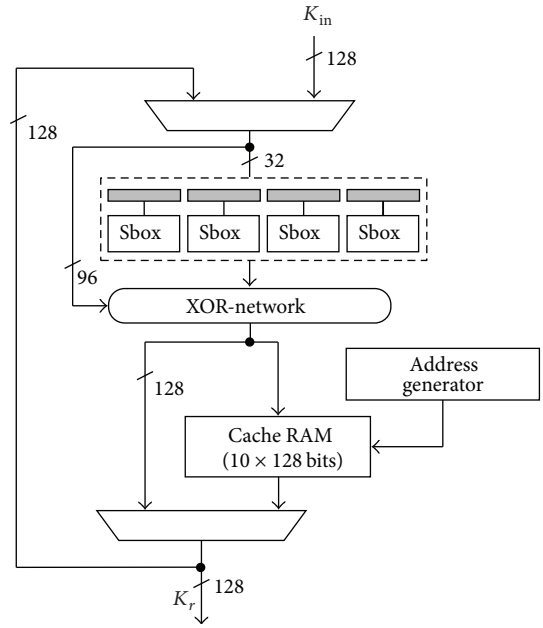


Figure 8: Key scheduler with caching mechanism.

successive counter encryptions, the round keys are read from the cache memory whereas the key computation mechanism is deactivated by the control unit. In addition to the caching mechanism, we perform register retiming by moving the registers at the output of the Sboxes to their input. This helps in eliminating the power consumption of Sboxes caused by logic hazards propagating from the feedback path through the input multiplexer.

*4.3. Round Shadowing.* According to the specification of the EEA2 algorithm [34], AES is not applied directly on user data blocks. It is however applied on incrementing counter blocks which are in turn mixed with the user data blocks to produce the cipher text. Figure 9 compares the state bytes of counter encryptions during the first and second encryption rounds. For each group of 256 counter blocks, only the least significant byte ($S_{15}$) is changed resulting in a change of a single column after the first round is completed. However, after MixColumns in the second round, all state bytes do not match any of their corresponding bytes in other counter block encryptions. Therefore, we identify three MixColumns and 24 Sbox redundant computations in the EEA2 algorithm during the first two rounds of each counter block encryption. For each PDCP packet with a typical length of thousands of bytes, equivalent to hundreds of block encryptions, thousands of redundant operations take place.

We exploit the redundant calculations in the first two rounds to reduce the power consumed in the Round block. Therefore, we insert a total of three 32-bit shadow registers distributed on the quarter rounds. Figure 10 shows the round shadowing architecture for the quarter rounds $QR_1$ to $QR_3$, where the shadow register is placed after the Sboxes. It is important to mention that we have reordered the ShiftRows and SubByte operations to maximize savings from redundant calculations. This modification allows us to save SubBytes computations of the second round at low hardware complexity without affecting the ciphering functionality. In addition, the control unit is adapted to activate the shadow registers after the first two rounds of the first counter block encryption. Thereby, the shadow registers store the state of the three columns resulting from the first two rounds. Afterwards the

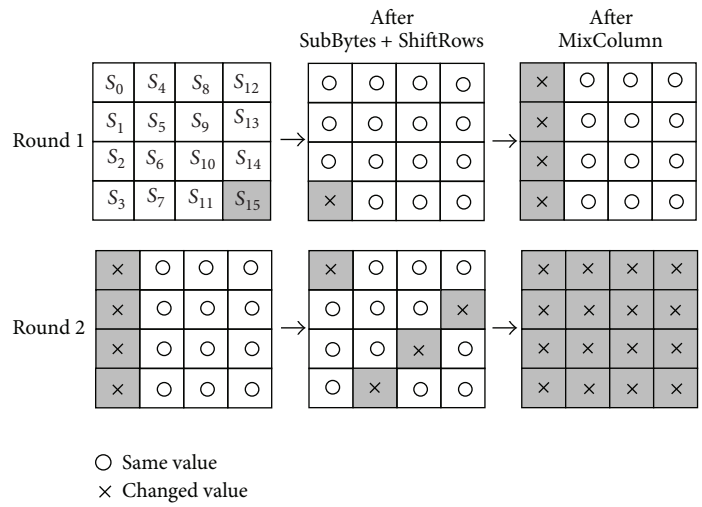


Figure 9: Demonstrating redundant computations in the encryption rounds of two consecutive counter blocks.

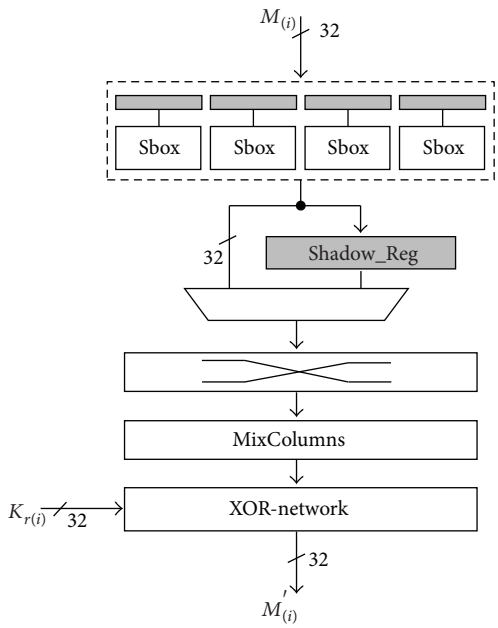


Figure 10: Quarter-Round architecture with a shadowing register.

control unit switches to the shadowing mode. In this mode, only $QR_0$ is enabled to compute the first column in the first round, whereas the other three columns are loaded from the shadow registers in the second round. The following rounds are processed with all quarter rounds being enabled. When the least significant byte in the counter block reaches a value of 255, the control unit switches back to the normal mode for one block encryption in order to update the value of shadow registers.

*4.4. Clock Gating.* To eliminate the dynamic power of idle components, clock gating [40] is applied on the entire 128-EEA2 hardware architecture. As the Round block stays in the shadowing mode for a large number of cycles, the shadow registers can be deactivated by inserting clock gating cells in order to avoid unnecessary internal power consumption. Similarly, the key cache in the key scheduler block can benefit from clock gating as it operates in the read mode for a long time. Therefore, some power can be saved by deactivating the clock of the huge 128-bit registers in the key scheduler. As a result, it is recommended to employ both key caching and round shadowing architectures associated with clock gating to optimize the design's power as intended. Furthermore, clock gating is applied for the counter block to deactivate its 32-bit register chunks which are not influenced by the incrementing process.

*4.5. Dynamic Data Path Adaptation.* This optimization targets power-constrained embedded systems. Thus, it aims at reducing the peak power consumption of the EEA2 core by adapting the data path width at runtime, according to the requested ciphering speed. Figure 11 shows a modified architecture of the Round block, where four 32-bit multiplexers are inserted to route state's columns to their respective quarter-round. The Round architecture supports three modes of operation: Full-Round (FR), Half-Round (HR), and Quarter-Round (QR) modes. In FR, all quarter-round modules are enabled to deliver full ciphering speed. Two quarter rounds ($QR_0$ and $QR_1$) are only enabled in the HR mode, whereas only a single round module ($QR_0$) is enable for the QR mode. The control unit is equipped with a finite state machine dedicated for configuring the routing multiplexers.

The data path mode can be switched dynamically at round level with a small delay of two cycles. Thus four 32-bit state registers ($SR_0$ to $SR_3$) are additionally required to buffer the block's *state* after each round to maintain synchronization between rounds operating at different data path modes. In complex systems, decisions in the control unit are typically based on notifications from a power controller unit which keeps track of system's workload behaviour and the charge level of device's energy buffer. The key scheduler is also slightly modified where the number of simultaneously

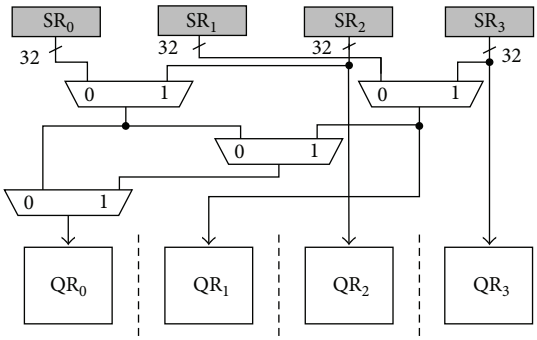


Figure 11: Round block architecture with support for dynamic data path adjustment.

enabled Sboxes is equal to that of the quarter rounds for a given data path mode. The proposed architecture also facilitates the application of power gating of quarter-round modules in order to reduce the leakage power which is a serious concern in deep submicron (65 nm and below) technology nodes [9].

## 5. LSB Coprocessor

The Window-based LSB (WLSB) encoding function in ROHC performs four LSB computations to encode each of the specified dynamic fields (IP_ID_DELTA and TS_SCALED) as described in Section 3.2. Therefore optimizing the LSB operation has a large impact on the function's performance and memory consumption. The window buffer in WLSB can be realized as a memory array in software, which is more practical as it provides the flexibility needed to reconfigure window sizes according to channel conditions. The software routine would also keep track of maximum and minimum reference values within a window by executing few instructions with relational operators. LSB computation is implemented by a dedicated hardware unit, which can be tightly attached as a coprocessor to the communication processor, that is, the CPU where the LTE protocol stack runs.

At least, two values ($v$ and $v_{\text{ref}}$) are required by the LSB coprocessor to compute the number of bits ($k$) to be transmitted, representing the compressed value of $v$. The value of $k$ as a function of $v$ and $v_{\text{ref}}$ can be derived from the interpretation interval as follows:

$$v_{\text{ref}} - p < v < v_{\text{ref}} + 2^k - 1 - p, \tag{3a}$$

$$k > lg\left(v - v_{\text{ref}} + 1 + p\right), \tag{3b}$$

$$k = \left\lceil lg\left(v - v_{\text{ref}} + 1 + p\right)\right\rceil. \tag{3c}$$

*5.1. Loop-Based Architecture.* Figure 12 depicts a loop-based architecture of the LSB coprocessor. For simplicity, we assume $p$ is equal to zero, that is, sequential behaviour. The hardware computes the exponent of the logarithmic function in (3c) with an adder and two subtractors. The second

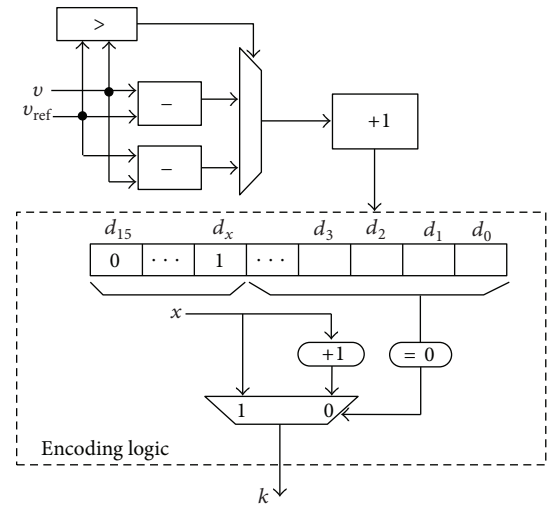


Figure 12: Loop-based architecture of the LSB coprocessor.

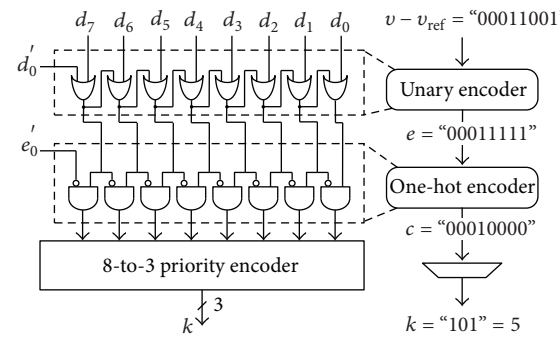


Figure 13: The proposed parallel-based architecture of the LSB coprocessor.

subtractor is needed to ensure the calculation of the absolute value of the difference ($v - v_{\text{ref}}$) in case $v$ has a decrementing behaviour and hence is smaller than its reference value. Afterwards, the value $k$ is computed by searching for the index ($x$) of the leftmost "1" bit in the exponent value, starting from the most significant bit (hence the loop behaviour). If any of the fractional (lower index) bits is high, then $k$ will be evaluated to $x$ incremented by one through a second adder logic, otherwise, it is set to $x$. This way of computation is inspired from the mathematical equations and hence the loop-based architecture can be considered as a direct or conventional LSB implementation [29]. The LSB coprocessor data path width can be adapted according to the size of the input word, which might vary from 8 to 64 bits.

*5.2. Parallel Architecture.* We propose a novel parallel-based architecture of the LSB coprocessor targeting a fast and low-power implementation of the LSB encoding algorithm. Figure 13 presents the structure of an 8-bit parallel LSB implementation. It contains three processing stages: unary encoding, one-hot encoding, and 8-to-3 bit priority encoding. The example shown in Figure 13 illustrates the operation of each processing stage. First, the difference ($d = v - v_{\text{ref}}$)

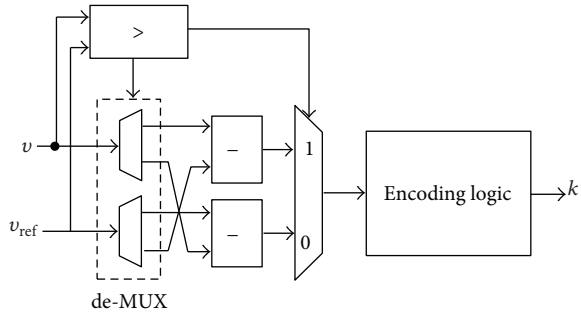


Figure 14: Architecture of the LSB coprocessor with operand isolation.

is computed before entering the unary encoder. The latter is composed of a chain of eight OR gates. The unary encoder output $e$ is fed into the next stage to produce a one-hot code $c$ highlighting the uppermost active bit in the difference $d$. Eight AND and eight NOT gates are used to realize the one-hot encoder. Finally, $k$ is produced by the priority encoder, which outputs the order of the active bit as a 3-bit binary word. The parallel bit implementation can be extended with other 8-bit chunks to form wider parallel architectures suitable for input words of various sizes. In this case, the inputs $d_0$ and $e_0$ are connected to their respective signals in the upper processing module. However, for 8-bit implementations those inputs are wired to zero.

By applying the encoding logic directly on the word difference, the parallel architecture is able to save the increment-by-one adder required in the loop-based implementation. In addition, the parallel architecture does not include any arithmetic component in its encoding logic compared to that of the loop architecture, where an adder is crucial for a correct functional behaviour. Moreover, the parallel architecture is expected to be fast as it provides a purely combinational realization of LSB versus the sequential behaviour of the loop-based implementation. Apart from that, the parallel LSB implementation is intentionally designed to reduce the rate of logic switching to a high extend through the one-hot encoder block.

*5.3. Operand Isolation.* Typically, two subtractors with a multiplexer are employed to calculate the absolute value of the difference $(v - v_{\text{ref}})$ as shown in Figure 12. Since the result of only one subtractor is eventually selected for further processing, we have one redundant subtraction every LSB encoding operation. Operand isolation technique can be applied as shown in Figure 14 to avoid switching both subtractors at a time. It deactivates the idle subtractor by inserting one demultiplexer (de-MUX) for each of the subtractor operands. The de-multiplexers are controlled by the comparator signal. For instance, if the first subtractor is selected, that is, $v > v_{\text{ref}}$, then both de-multiplexers isolate the operands from the second subtractor by assigning its inputs to null.

## 6. Results

The new architectures proposed in the previous sections are described in VHDL models and synthesized by Synopsys' DesignVision Version B-2008.09 [41]. Synthesis is based on Faraday's standard cell library of 90 nm CMOS technology with a core power supply of 1 V. Post-synthesis simulations are performed in ModelSim, where the EEA2 and LSB architectures are verified against 3GPP specifications. The EEA2 core is driven by 1000 bytes randomly generated PDCP packets, whereas the LSB coprocessor is supplied with random values ($v$ and $v_{\text{ref}}$) adapted to the target data path width. The switching activity in designs' netlist is recorded and adopted for a subsequent accurate power analysis based on realistic power estimations. When referring to area efforts, the values are expressed in kilo Gate Equivalents (kGE), which is computed through dividing the total area by the size of a 2-input AND gate ($5\,\mu\text{m}^2$) in the adopted technology library. Results reported in the following paragraphs are obtained by the comparison of the performance of the optimized proposed architecture against a straight-forward or conventional reference architecture.

*6.1. 128-EEA2 Crypto-Core.* Synthesis results show that the crypto-core supports a maximum frequency of 365 MHz with a block ciphering latency of 11 cycles. Therefore, it is capable of providing a processing speed of 4.25 Gbit/s, a value that goes well beyond the currently defined LTE-Advanced requirements. The performance can be easily improved by applying conventional techniques such as pipelining. However, we do not employ such techniques as our main goal is not to achieve the fastest possible design, but rather to provide a processing speed which is high enough to support LTE-Advanced and beyond communication standards while implementing techniques for power and area reduction. In the following, we will use both terms *processing time-per-byte* and *processing speed* to refer to the performance of the hardware. The processing speed can be calculated by dividing the processing time-per-byte by the number of cycles-per-byte. In our discussions, we focus on three speed categories: time-per-byte of 16 ns, corresponding to 500 Mbit/s that is, the higher limit of the LTE requirements, time-per-byte of 8 ns, corresponding to 1 Gbit/s, that is, within the LTE-Advanced definition, and finally time-per-byte of 4 ns corresponding to 2 Gbit/s, that is, in the beyond-LTE-Advanced domain.

Figure 15 shows the power consumption of the EEA2 core based on several Sbox architectures. As the processing time per byte reduces, the clock frequency increases exponentially, hence the power consumption. Maximum power savings of 29% is achieved by DSE due to its one-hot encoding which heavily reduces the switching rate in the Sbox. NBDSE comes second with a slightly lower power reduction (2%) compared to DSE. This is mainly due to the extra switching activity in the design subswitches. Finally, CBI comes third in place with power reduction of 21%. For very low processing time per byte (6 ns), the percentage of power savings is increased up to 38% with DSE since the critical path in LUT is longer than that in other architectures. Therefore, more resources and power are consumed to satisfy the target processing speed. This critical path effect can however be eliminated by inserting pipeline stages which shorten design's critical paths.

Figure 16 shows a comparison in the area consumed by the EEA2 core based on different Sbox architectures.

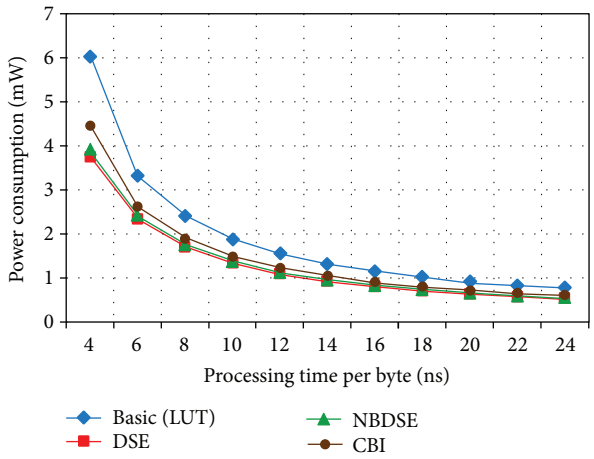


Figure 15: Power consumption of the 128-EEA2 core with different Sbox approaches compared to the look-up table (LUT) implementation.

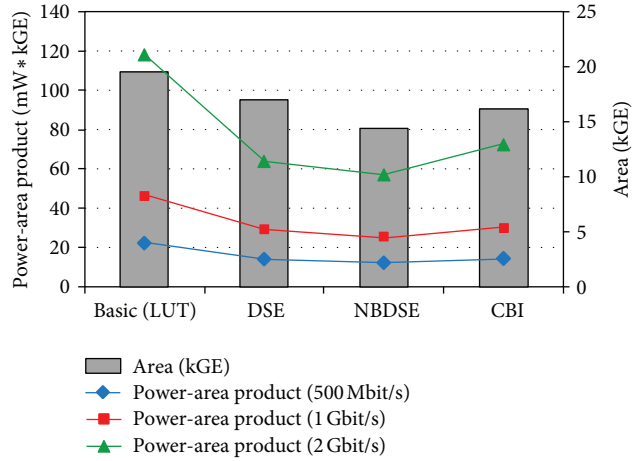


Figure 16: Area consumption and power-area product of the 128-EEA2 core for different Sbox approaches compared to the look-up table (LUT) implementation.

NBDSE outperforms other architectures with a maximum area reduction of 26%. This huge saving in area is caused by splitting the byte transformation into two nibble transformations, hence reducing the complexity of the output encoders. CBI achieves an area reduction of 17%, which is basically limited by the relatively large area consumed by row multiplexers. Due to its large 256-to-8 bit output encoder, DSE delivers the lowest area reduction of 13%. To make a fair evaluation with respect to design efficiency, we consider the power-area product which takes both area and power dimensions into consideration. Figure 16 shows the power-area product of the candidate architectures for several processing speeds. NBDSE is ranked first with and an average power-area product reduction of 46%. DSE and CBI follow with product reductions of 38% and 36%, respectively. For 2 Gbit/s, the power-area product loses its scalability observed in lower speeds due to the critical path effect mentioned earlier. Nevertheless, NBDSE achieves further reductions reaching 52%. As a result, we identify the nibble-based decoder-switch-encoder design as the most efficient among other architectures.

Next, we investigate the impact of key caching and round shadowing optimizations on both power and area consumption. As depicted in Figure 17, the key caching (KC) technique achieves up to 25% of power savings. Here the cipher key is set to change once every 50 block encryptions. Power reductions obtained through the caching mechanism comes however at the expense of a huge area overhead of up to 30%, caused by the inserted cache memory. However, round shadowing (RS) reduces power by 12% with its intelligent exploitation of redundant round computations. Although three shadow registers are added to the design, the area of EEA2 based on RS is reduced by 12%. As clock gating is associated with round shadowing, this area reduction is justified by the use of clock gating cells which are efficiently designed and therefore eliminate the cost of registers' enable multiplexers. By combining both RS and NBDSE optimizations, we are able to achieve 35% reductions in power consumption and area

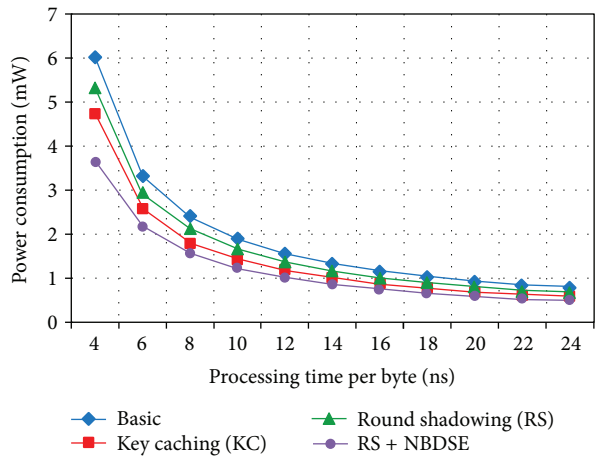


Figure 17: Power consumption of the 128-EEA2 core for different optimization approaches compared to the basic implementation.

savings up to 25%. With this architectural combination the power of Sboxes is heavily reduced. It has been illustrated in Section 4 that the switching activity of Sboxes in the key scheduler is much higher than that in the round block. Thus, by applying KC together with RS and NBDSE, further savings in power consumption will be negligible. By considering the power-area product depicted in Figure 18, we observe that KC provides only a slight (4%) product reductions at processing speeds below 1 Gbit/s, whereas it results in a slight (2%) product increase at processing speeds of 2 Gbit/s due to an increase in the occupied area. However, RS achieves 24% of power-area product reduction and even increases it up to 51% when applied together with NBDSE. As a result, we identify both round shadowing and nibble-based one-hot-encoding mechanisms as the most efficient for obtaining a low-power and compact 128-EEA2 hardware unit.

Figure 19 depicts the power consumption of the 128-EEA2 architecture with support for dynamic data path

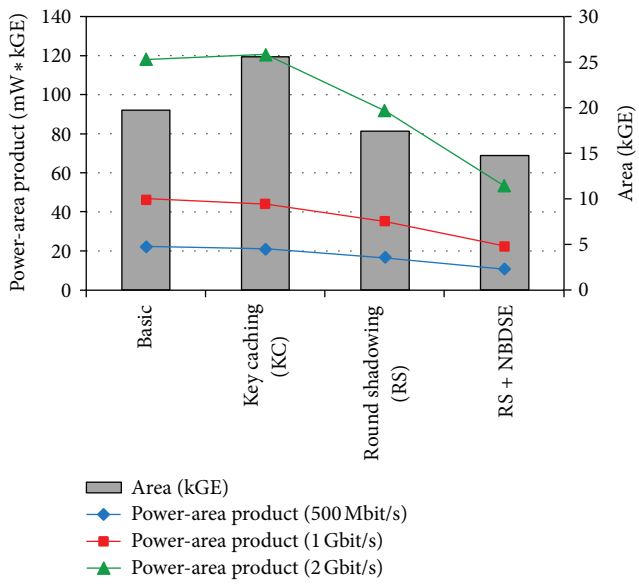


FIGURE 18: Area consumption and power-area product of the 128-EEA2 core for different optimization approaches compared to the basic implementation.

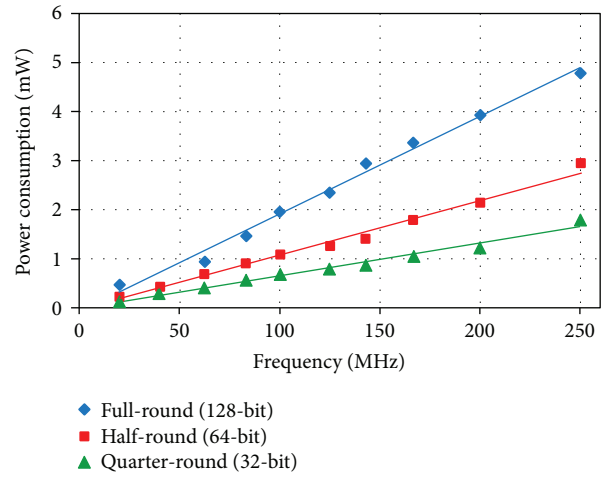


FIGURE 19: Power consumption of the 128-EEA2 core with data path adjustment architecture for different round modes.

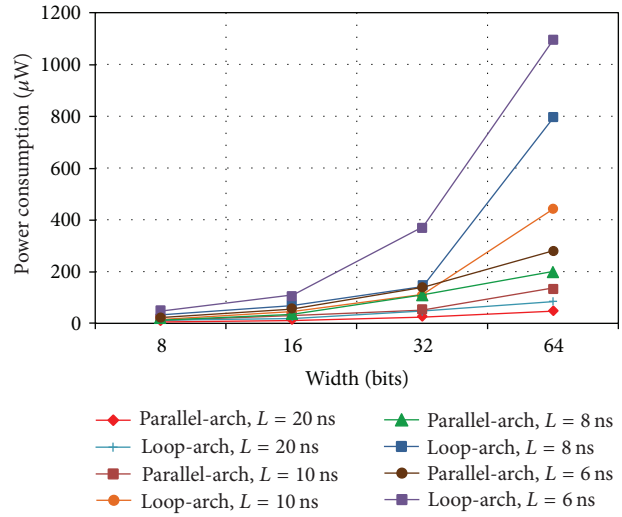


FIGURE 20: Power consumption of the LSB coprocessor with parallel and loop implementations.

adjustment. Here Sboxes are implemented with NBDSE architecture. When operating in Half-Round mode, the peak power consumption is reduced by 53%. However the peak power can be further reduced by 71% when the Quarter-Round mode is activated. This architecture is quite useful, for example, for power-constrained energy-harvesting devices, such as modern mobile devices which might partly rely on solar energy to charge their batteries and hence can keep alive even during critical battery lifetime levels. The hardware would switch to low data path modes if the target data rate is relatively low, depending on the requested service. Alternatively, the data path can be adjusted to limit the peak power consumed by the design depending on the amount of charge available in the energy buffer of the device. In both scenarios, this optimization enables such devices to live longer and work properly at lower quality of service rather than causing equipment's death and hence complete loss of communication. The proposed architecture is also capable to switch between modes rapidly with a fine, round-level granularity. Therefore, it has a major advantage over other techniques such as voltage and frequency scaling, which typically require design resetting and reinitialization mechanisms.

*6.2. LSB Coprocessor.* Both parallel and loop LSB architectures are investigated at various word sizes or architectural widths. For 16-bit architectures, the loop implementation supports a maximum frequency of 230 MHz with a corresponding throughput of 4.6 Gbit/s. However, the parallel implementation achieves a maximum clock speed of 536 MHz which is equivalent to 8.4 Gbit/s. Therefore, our proposed architecture is almost twice as fast as the conventional implementation. By adding a pipelining register at the input of the encoding logic, the processing throughput of the parallel implementation can be raised up to 12.6 Gbit/s.

Figure 20 shows the power consumption of the LSB coprocessor unit with parallel and loop implementations configured for encoding of 8-, 16-, 32-, and 64-bit word sizes. To analyze the impact of processing time on power consumption, we have considered several processing latencies ($L$), ranging from 20 ns down to 6 ns. It is obvious that the parallel implementation (Parallel-arch) is more power efficient than the loop implementation (Loop-arch) as it consumes around 50% less power for 8- and 16-bit architectures. This is due to the low switching rate provided by the unary and one-hot encoding blocks. The loop architecture increases rapidly with short processing times because of its longer critical path compared to Parallel-arch. Moreover, Loop-arch is not scalable with larger bit widths as it is realized using a chain of cascaded multiplexers which increases with the size of the input word. Therefore, larger power savings of 62% and 74% are observed for the Parallel-arch at 32- and 64-bit

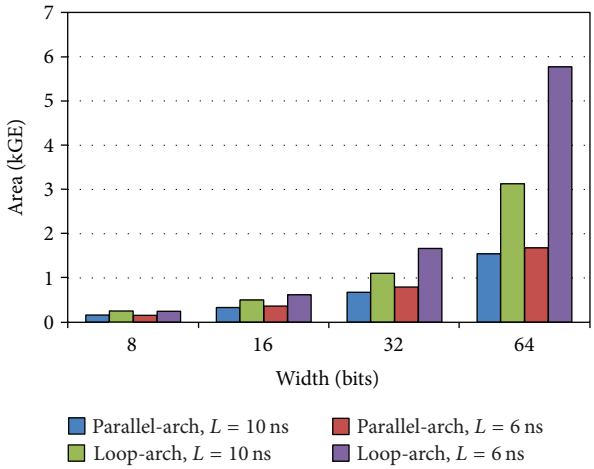


FIGURE 21: Area consumption of the LSB coprocessor with parallel and loop implementations.

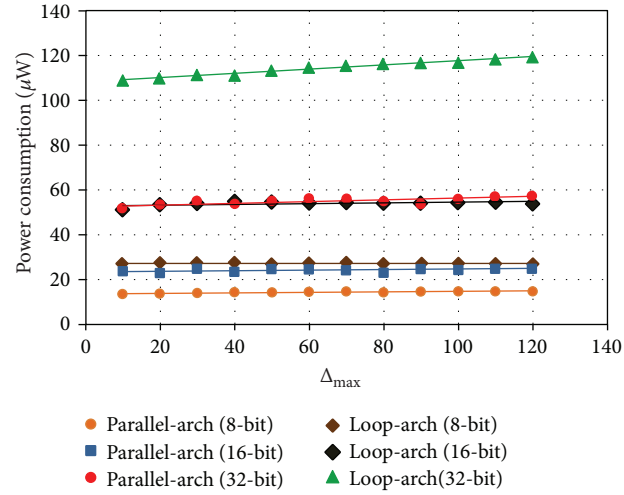


FIGURE 22: Impact of the word difference Δ on the power consumption of the LSB coprocessor with parallel and loop implementations.

architectures, respectively. However, the power consumption curves of Parallel-arch have a linear behavior and hence are scalable with respect to the word size and the processing latency.

Figure 21 depicts the area consumption of the LSB coprocessor unit with parallel and loop implementations. Results show that the proposed parallel architecture is more compact than the loop architecture and has a linear increase in area consumption with respect to word size. However, the area of the loop implementation increases exponentially with the word size, which has a strong impact on the length of its critical path. Thus, the area efficiency of Parallel-arch compared to Loop-arch increases with the word size. In particular, area savings start with 36% in 8-bit architectures and goes up to 71% for 64-bit architectures. Besides, average area reductions of 41% and 52% are obtained at 16-bit and 32-bit architectures, respectively.

As the difference between the word to be encoded and its reference value forms the input to the encoding logic, we study the impact of this difference (Δ) on the power consumption of both parallel and loop architectures. For this purpose, the test bench is adapted to limit the difference value to an upper bound of $\Delta_{max}$. Figure 22 depicts the power consumption of the LSB coprocessor with respect to $\Delta_{max}$. As the range of Δ increases, more logic gates are switched throughout the architecture causing a higher power to be consumed. While the raise in power consumption is negligible at 8-bit and 16-bit architectures, it can be clearly observed with 32-bit architectures. This can be distinguished from the slopes of the linear power consumption curves. It can be noticed that the slope of the power curves for the proposed Parallel-arch is smaller than that of the Loop-arch. Figure 22 shows a deviation of 9% in power values of the 32-bit Loop-arch which is roughly twice that of the Parallel-arch. Such an interesting behaviour indicates that the parallel architecture has low power fluctuations over its operation time and hence can be easily integrated in high-level power estimation tools [42]. Moreover, Parallel-arch

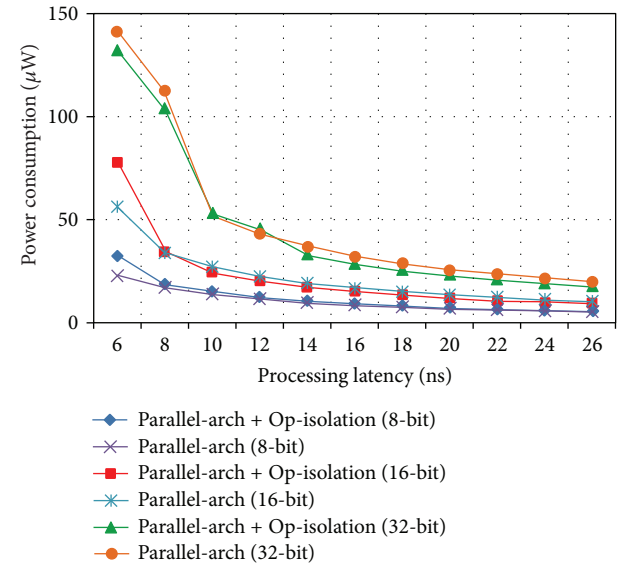


FIGURE 23: Impact of operand isolation on the power consumption of the LSB coprocessor.

obtains even higher power reductions, compared to Loop-arch, for increasing values of Δ.

Next, we investigate to what limit operand isolation can reduce the power consumed by the LSB coprocessor. Here, operand isolation is combined with the parallel architecture, as the latter has proved to be the most efficient in terms of power and area cost. Figures 23 and 24 depict the power and area consumption for the parallel implementation in its both versions, that is, with and without operand isolation (Op-Isolation). For 8-bit architectures, no power reduction is achieved by operand isolation, because the additional power consumed in the de-multiplexer and the encoding logic (due to logic hazards) compensates the power savings obtained by the subtractors. However, wider architectures with operand isolation achieves power savings of 10%, equivalent to 20%

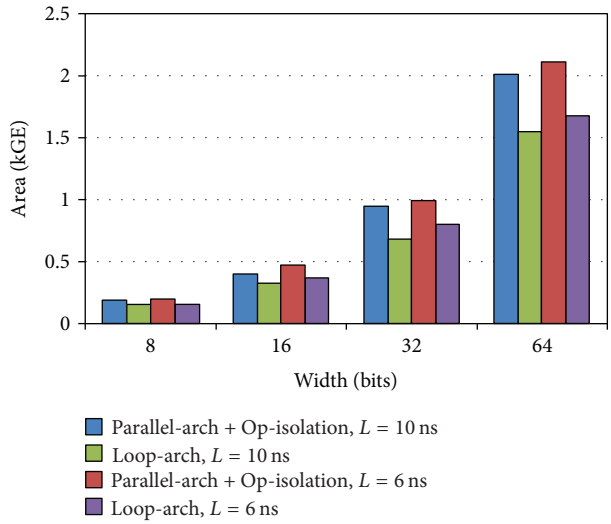


Figure 24: Impact of operand isolation on the area consumption of the LSB coprocessor.

with respect to Loop-arch, at processing latencies below 10 ns. Actually by isolating the subtractor operands we are increasing the critical path of the circuit though the demultiplexer logic. This causes more glitches to appear at the input of the encoding logic causing more area and power to be consumed at very low processing latencies, that is, below 8 ns. This effect however can be eliminated using pipelined architectures, where the critical path can be reduced and the target latency can be fulfilled without additional synthesis effort. A clear drawback of operand isolation is shown in Figure 24, where an average area overhead of 25% is obtained. As a result, operand isolation might only be recommended for applications with hard power constraints, that is, where the primary goal is to achieve the lowest power consumption possible.

## 7. Conclusion

Upcoming mobile devices have to perform fast packet processing in order to support high data rates up to 1 Gbit/s in LTE-Advanced networks. In addition, implementations of packet processing functions must possess low energy consumption profiles in order to increase handset's battery lifetime. In this paper, we have presented and evaluated energy-efficient cores to accelerate ciphering (128-EEA2) and header compression (ROHC) functions in the Layer 2 PDCP sublayer of a cellular protocol stack. In particular, we investigate the 128-EEA2 confidentiality algorithm that is based on AES cipher. For this purpose, we have introduced and explored several architectures for optimizing the power of AES substitution boxes and increasing its compactness. In addition, we introduce power reductions at the algorithmic level through round shadowing by eliminating redundant computations in the first two AES rounds within an encryption of a counter block. Similarly, key caching is applied to restrict the computation of round keys to situations where the cipher key is reassigned by the radio resource control. We supported key caching and round shadowing with clock gating to eliminate the internal power of idle components, such as shadow and key cache registers. A special architecture of 128-EEA2 was also proposed and tailored for energy harvesting devices with hard peak power constraints. A reconfigurable data path architecture is developed for this purpose to control the peak power consumption at run time, however at the expense of the quality of service. However, the second core supporting header compression implements an LSB coprocessor with a power-efficient parallel architecture based on a one-hot encoding. Hardware architectures have been implemented and synthesized for a 90 nm ASIC technology. Results show maximum power (35%) and area (25%) reductions for the 128-EEA2 core associated with an architecture combining round shadowing, nibble-based Decoder-Switch-Encoder substitution boxes, and clock gating. Furthermore, the proposed parallel implementation for the LSB coprocessor has shown power and area reductions starting from 50% and 36%, respectively.

## References

[1] The 3rd Generation Partnership Project (3GPP), http://www.3gpp.org.

[2] D. Szczesny, S. Hessel, A. Showk, A. Bilgic, U. Hildebrand, and V. Frascolla, "Performance analysis of LTE protocol processing on an ARM based mobile platform," in *Proceedings of the 11th International Symposium on System-on-Chip (SoC '09)*, pp. 56–63, Tampere, Finland, October 2009.

[3] D. Szczesny, S. Hessel, A. Showk, A. Bilgic, U. Hildebrand, and V. Frascolla, "Joint uplink and downlink performance profiling of LTE protocol processing on a mobile platform," *International Journal of Embedded and Real-Time Communication Systems*, vol. 1, no. 4, pp. 21–39, 2010.

[4] ARM Limited, "ARM11 MPCore Processor Technical Reference Manual", Lit.Nr.: ARM DDI, 0360D, 2006.

[5] M. E. Gonzalez, A. Bilgic, A. Lackorzynski, D. Tudor, E. Matus, and I. Badr, "ICT-eMuCo. An innovative solution for future smart phones," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '09)*, pp. 1821–1824, New York, NY, USA, July 2009.

[6] A. Showk, F. Bruns, S. Hessel, A. Bilgic, and I. Badr, "Optimal resource management for a model driven LTE protocol stack on a multicore platform," in *Prpceedings of the ACM International Symposium on Mobility Management and Wirelss Access (MobiWac '10)*, pp. 91–98, Bodrum, Turkey, October 2010.

[7] A. Showk, S. Traboulsi, D. Szczesny, and A. Bilgic, "Balancing LTE protocol load on a multi-core hardware platform using EFSM migration," in *Proceedings of the 6th International Multi-Conference on Computing in the Global Information Technology (ICCGI '11)*, pp. 76–83, Luxembourg, Germany, June 2011.

[8] M. Hill and M. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, no. 7, pp. 33–38, 2008.

[9] J. Rabaey, *Low Power Design Essentials*, Springer, New York, NY, USA, 2009.

[10] M. Ouellette and D. Connors, "Analysis of hardware acceleration in reconfigurable embedded systems," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05)*, Denver, Colo, USA, April 2005.

[11] C. H. van Berkel, "Multi-Core for mobile phones," in *Proceedings of the Design, Automation & Test Europe (DATE '09)*, pp. 1260–1265, Nice, France, April 2009.

[12] O. Silven and K. Jyrkk, "Observations on power-efficiency trends in mobile communication devices," *EURASIP Journal of Embedded Systems*, vol. 2007, no. 1, 17 pages, 2007.

[13] S. Hessel, D. Szczesny, S. Traboulsi, A. Bilgic, and J. Hausner, "On the design of a suitable hardware platform for protocol stack processing in LTE terminals," in *Proceedings of the 7th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC '09)*, pp. 1–8, Vancouver, Canada, August 2009.

[14] D. Szczesny, S. Hessel, F. Bruns, and A. Bilgic, "On-the-fly hardware acceleration for protocol stack processing in next generation mobile devices," in *Proceedings of the 5th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '09)*, pp. 155–162, Grenoble, France, October 2009.

[15] S. Hessel, D. Szczesny, F. Bruns, A. Bilgic, and J. Hausner, "Architectural analysis of a smart DMA controller for protocol stack acceleration in LTE terminals," in *Proceedings of the IEEE 72nd Vehicular Technology Conference Fall (VTC '10)*, pp. 1–5, Ottawa, Canada, September 2010.

[16] P. Kitsos, G. Selimis, and O. Koufopavlou, "High performance ASIC implementation of the SNOW 3G stream cipher," in *Proceedings of the International Conference on Very Large Scale Integration (VLSI-SOC '08)*, Rhodes Island, Greece, October 2008.

[17] P. Kitsos, N. Sklavos, and A. N. Skodras, "An FPGA implementation of the ZUC stream cipher," in *Proceedings of the 14th Euromicro Conference on Digital Systems (DSD '11)*, Oulu, Finland, August 2011.

[18] Z. Liu, L. Zhang, J. Jing, and W. Pan, "Efficient pipelined stream cipher ZUC algorithm in FPGA," in *Proceedings of the 1st International Workshop on ZUC Algorithm*, Beijing, China, December 2010.

[19] S. Traboulsi, V. Frascolla, N. Pohl, J. Hausner, and A. Bilgic, "Power analysis and optimization of the ZUC stream cipher for LTE-advanced mobile terminals," in *Proceedings of the 3rd IEEE Latin American Symposium on Circuits and Systems (LASCAS '12)*, Playa del Carmen, Mexico, February 2012.

[20] S. Traboulsi, V. Frascolla, N. Pohl, J. Hausner, and A. Bilgic, "A versatile low-power ciphering and integrity protection unit for LTE-advanced mobile devices," in *Proceedings of the 10th IEEE International NEWCAS Conference (NEWCAS '12)*, Montreal, Canada, June 2012.

[21] S. Tillich, M. Feldhofer, T. Popp, and J. Großschädl, "Area, delay, and power characteristics of standard-cell implementations of the AES S-box," *Journal of Signal Processing Systems*, vol. 50, no. 2, pp. 251–261, 2008.

[22] S. Morioka and A. Satoh, "An optimized S-Box circuit architecture for low power AES design," in *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '02)*, vol. 2523 of *Lecture Notes in Computer Science*, pp. 172–186, 2002.

[23] G. Bertoni, M. Macchetti, and L. Negri, "Power-efficient ASIC synthesis of cryptographic sboxes," in *Proceedings of the 14th ACM Great Lakes Symposium on VLSI (GLSVLSI '04)*, pp. 277–281, April 2004.

[24] S. Hessel, D. Szczesny, N. Lohmann, A. Bilgic, and J. Hausner, "Implementation and benchmarking of hardware accelerators for ciphering in LTE terminals," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '09)*, Honolulu, Hawaii, USA, December 2009.

[25] S. Traboulsi, M. Sbeiti, D. Szczesny, A. Showk, and A. Bilgic, "High-performance and energy-efficient sliced AES multi-block encryption for LTE mobile devices," in *Proceedings of the International Conference on Computer and Communication Devices (ICCCD '11)*, Bali, Indonesia, April 2011.

[26] S. Traboulsi, M. Sbeiti, F. Bruns, S. Hessel, and A. Bilgic, "An optimized parallel and energy-efficient implementation of SNOW 3G for LTE mobile devices," in *Proceedings of the IEEE 12th International Conference on Communication Technology (ICCT '10)*, pp. 535–538, Nanjing, China, November 2010.

[27] A. Showk, S. Traboulsi, and A. Bilgic, "An energy efficient multi- core modem architecture or LTE mobile terminals," in *Proceedings of the The New Technologies, Mobility and Security Conference (NTMS '12)*, Istanbul, Turkey, May 2012.

[28] D. Szczesny, S. Traboulsi, F. Bruns, S. Hessel, and A. Bilgic, "Acceleration concepts for the ROHCv2 in LTE handsets," in *Proceedings of the 6th International Symposium on Industrial Embedded Systems (SIES '11)*, Vasteras, Sweden, June 2011.

[29] D. E. Taylor, A. Herkersdorf, A. Döring, and G. Dittmann, "Robust header compression (ROHC) in next-generation network processors," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 755–768, 2005.

[30] S. Traboulsi, W. Zhang, D. Szczesny, A. Showk, and A. Bilgic, "An energy-efficient hardware accelerator for robust header compression in LTE-advanced terminals," in *Proceedings of the 22nd International Conference on Field Programmable Logic and Applications (FPL '12)*, Oslo, Norway, August 2012.

[31] K. Lahiri and A. Raghunathan, "Power analysis of system-level on-chip communication architectures," in *Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '04)*, pp. 236–241, Stockholm, Sweden, September 2004.

[32] G. Stitt, R. Lysecky, and F. Vahid, "Dynamic hardware/software partitioning: a first approach," in *Proceedings of the 40th Design Automation Conference (DAC '03)*, pp. 250–255, Anaheim, Calif, USA, June 2003.

[33] "Evolved universal terrestrial radio access (E-UTRA): packet data convergence protocol (PDCP) specification, 3GPP Std," TS 36.323, Rev. 8.6.0, 2009.

[34] "GPP system architecture evolution (SAE): security architecture, 3GPP Std," TS 33.401, Rev. 8.2.1, 2008.

[35] National Institute of Standards Technology, "NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation—Methods and Techniques," December 2001.

[36] National Institute of Standards Technology, FIPS Publication 197: Advanced Encryption Standard (AES), November 2001.

[37] C. Bormann, C. Burmeister, M. Degermark et al., "Robust header compression (ROHC): framework and four profiles: RTP, UDP, ESP, and uncompressed," RFC 3095, 2001.

[38] The Internet Engineering Task Force (IETF), http://www.ietf.org.

[39] G. Pelletier and K. Sandlund, "Robust header compression version 2 (ROHCv2): profiles for RTP, UDP, IP, ESP and UDP-Lite," RFC 5225, 2008.

[40] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Computing Surveys*, vol. 37, no. 3, pp. 195–237, 2005.

[41] Synopsys Inc, http://www.synopsys.com.

[42] C. Talarico, J. V. Rosenblit, V. Malhotra, and A. Sitter, "A new framework for power estimation of embedded systems," *Computer*, vol. 38, pp. 71–78, 2005.