Low-Latency Encryption – Is "Lightweight = Light + Wait"?*

Miroslav Knežević, Ventzislav Nikov, and Peter Rombouts

NXP Semiconductors, Leuven, Belgium

Abstract. The processing time required by a cryptographic primitive implemented in hardware is an important metric for its performance but it has not received much attention in recent publications on lightweight cryptography. Nevertheless, there are important applications for cost effective low-latency encryption. As the first step in the field, this paper explores the low-latency behavior of hardware implementations of a set of block ciphers. The latency of the implementations is investigated as well as the trade-offs with other metrics such as circuit area, time-area product, power, and energy consumption. The obtained results are related back to the properties of the underlying cipher algorithm and, as it turns out, the number of rounds, their complexity, and the similarity of encryption and decryption procedures have a strong impact on the results. We provide a qualitative description and conclude with a set of recommendations for aspiring low-latency block cipher designers.

1 Introduction

As cryptography is becoming ever more pervasive in modern technology, new applications regularly emerge. Some of these new applications also introduce new requirements on the implementation such as ultra fast response times. Applications such as Car2X communication (e.g. automotive road tolling, intelligent transport systems), high speed networking (optical links), and secure storage devices (e.g. memories, solid-state disks, super-speed USB 3.0), just to name a few, all require an instant response. Besides these there are also applications that require moderately high throughput but have limited maximum clock frequencies, e.g. FPGA, or strict area requirements that preclude the use of highly pipelined architectures.

Cryptographic primitive design is a balancing act between several aspects such as cryptographic strength, implementation cost, execution speed, power consumption, etc. Which trade-offs are the right ones to make is determined by

* The authors would like to thank Bruce Murray for his valuable comments and suggestions to improve the work. The authors also thank Hans De Kuyper for his valuable inputs. The work has been supported by the European Commission through the ICT program under contract ICT-2007-216646 (European Network of Excellence in Cryptology – ECRYPT II), through the Tamper Resistant Sensor Node (TAMPRES) project with contract number 258754 and through the Internet of Things - Architecture (IoT-A) project with contract number 257521.

E. Prouff and P. Schaumont (Eds.): CHES 2012, LNCS 7428, pp. 426–446, 2012.

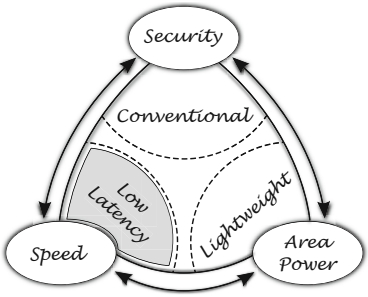

Fig. 1. Typical trade-offs in cryptography

the application. In the past, different applications have led to different corners of the design space to be explored. The most important of these are depicted in Fig. 1.

Government applications have typically favored cryptographic strength over aspects such as cost and speed, although these aspects usually do play an important role in selection processes like the former AES competition [1] and currently the SHA-3 competition [31]. The use of these algorithms in applications such as mainframe systems has resulted in the development of high throughput implementations, both in hardware and software.

More recently the advent of RFID and other wireless technologies sparked an interest in a new field: low-power and low-cost cryptography. The first primitives to be explored were stream ciphers, for example in the eSTREAM project [15], followed by a whole range of block ciphers such as TEA [37], NOEKEON [13], MINI-AES [12], MCRYPTON [29], SEA [36], HIGHT [23], DESXL [27], CLEFIA [35], PRESENT [9], MIBS [24], KATAN/KTANTAN [10], PRINTcipher [26], KLEIN [18], LED [20], PICCOLO [34], and others. The field has recently been expanded by the introduction of several new low-cost hash functions such as DM-PRESENT [20], KECCAK-f[400]/-f[200] [7, 25], QUARK [6], PHOTON [19], and SPONGENT [8].

We have identified a new range of applications; those that require very fast response times and for which there is no established research field yet. Note that although most of the high-speed implementations available in literature do achieve tremendous throughput, their response time is generally not that fast. This is due to their extensive use of pipelining which enables them to process multiple messages at the same time, but in order to encrypt a single message block, this type of implementation still needs multiple clock cycles, i.e. typically more than 20. An example of this is a recent work from Mathew et al. [30], presenting a reconfigurable AES encrypt/decrypt hardware accelerator targeted for content-protection in high-performance microprocessors which, manufactured in 45 nm CMOS technology, achieves 53 Gb/s throughput. Another example comes from Hodjat and Verbauwhede [22] where area-throughput trade-offs of a fully pipelined AES implementation are described and a throughput of 30 Gb/s to 70 Gb/s is achieved.

In other words, a high throughput is usually achieved by common signal processing techniques such as pipelining and parallel processing, while achieving

a low latency, on the other hand, still remains a challenge. As a consequence one could ask the following questions: What is the minimum achievable latency for a given security level? Do designs that inherently have lower latency also achieve higher throughput when implemented in a pipelined fashion? And does "lightweight" necessarily mean "light + wait?" These are all interesting questions and as it seems there are a lot of compelling reasons to take a closer look at the latency behavior of cryptographic primitives.

Our Contribution. We introduce the new field of low-latency encryption; highlight the differences with lightweight and classical cryptography, and by bringing several important applications to light we try to motivate further research in this field.

We identify several well-known lightweight block ciphers as possible candidates to yield low-latency implementations. By examining this set of ciphers in the context of low-latency encryption, our work provides the first results in the field. We therefore develop a framework that examines the low-latency behavior of cryptographic primitives on the following aspects:

- Minimal achievable latency.
- Its impact on the circuit size.
- Its impact on the power and energy consumption.

We link the collected data to the cipher design decisions and show that results are strongly influenced by their properties. More specifically, the number of rounds, the round's complexity (e.g. the S-box size, MDS (Maximum Distance Separable) matrices defined over different fields versus binary matrix), and the similarity of encryption and decryption procedures have a significant influence on the algorithm's performance. Our work concludes with a set of recommendations for aspiring low-latency block cipher designers.

Organization of the Paper. The remainder of this paper is organized as follows. In Section 2, we provide a short description of the block ciphers we have chosen to investigate. Our contributions – the implementation results, comparisons, and discussion – are presented in Sections 3 and 4. We first investigate the minimum achievable latency in Section 3.1 and then evaluate the impact optimization for low latency has on area in Section 3.2. Our study continues by combining the two previously described metrics in section Section 3.3 where the results for the time-area product are presented and in Section 3.4, we have a closer look at the impact low-latency implementations have on the power and energy consumption. We elaborate more on our results and conclude in Section 4.

2 Preliminaries

There are many algorithms to choose from for a comparative study of low-latency behavior, but in order to draw meaningful conclusions about hardware

performance a set of candidate algorithms should be chosen with similar properties. We therefore focus on algorithms that are expected to result in low-latency implementations. Since hardware implementations of hash functions generally require more area to implement [16] and stream ciphers usually need a large number of initialization rounds [11, 21] we chose to focus on block ciphers only. Furthermore, it is expected that lightweight block ciphers yield good results in terms of implementation cost, even in a fully-unrolled implementation. Besides latency as our primary goal, we consider silicon area as a very important factor in practical implementations of encryption algorithms and, therefore, we restrict our candidates to lightweight block ciphers but include AES as the reference cipher. In order to reduce the number of candidates to a manageable number, we further restrict the set to ciphers with the well-studied SPN structure.

This results in the following list of seven lightweight SPN block ciphers: AES [14,32], KLEIN [18], LED [20], MCRYPTON [29], MINI-AES [12], NOEKEON [13], PRESENT [9]. We provide a brief description of each cipher and refer for more details to their original descriptions in the literature.

AES [14,32], designed by Daemen and Rijmen in 1997, has become not only a NIST standard but also the most used block cipher nowadays. The cipher has not been considered lightweight until the work of Feldhofer et al. [17] who provided the smallest implementation at the time, requiring only 3400 GE.[1] AES is an iterated block cipher with a block-size of 128 bits and three possible key lengths of 128, 192, and 256 bits. In this work, we consider only the 128-bit key version which consists of 10 rounds. The word size is 8 bits, i.e. the data elements are considered as elements of the field GF(2^8). Each round of AES consists of the following operations: *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. The operation *SubBytes* (S-layer) is defined as the simultaneous application of the S-Box (inversion in GF(2^8)) to each element of the state. The permutation layer (P-layer) consists of *ShiftRows* and *MixColumns* operations. The *ShiftRows* operation is defined as the simultaneous left rotation of the row i of the state by i positions. The *MixColumns* operation pre-multiplies each column of the state by an MDS matrix defined over GF(2^8). The *KeySchedule* derives the round key from the secret key, by applying once the S-Box and some simple linear operations. Finally, *AddRoundKey* XORs the round key to the current state.

NOEKEON [13] is a 128-bit block cipher with a 128-bit key, proposed by Daemen, Peeters, Van Assche, and Rijmen in 2000. NOEKEON is a self-inverse, bit-sliced cipher and can be considered as the predecessor of modern lightweight block ciphers. It has 16 rounds and each of them consists of the following operations: *Theta*, *Pi1*, *Gamma*, and *Pi2*. The operation *Gamma* is an involutive non-linear mapping (S-layer), in which S-boxes operate independently on 32 4-bit tuples. *Pi1* and *Pi2* perform simple cyclic shifts. *Theta* is a linear mapping that first XORs the working key to the state and then performs a simple linear transformation of the state. Therefore, *Theta* acts partially as *AddRoundKey*

[1] The current smallest implementation of AES comes from Poschmann et al. [33] and consumes only 2400 GE, which is comparable to the size of some of the first proposed lightweight block ciphers.

and, together with *Pi1* and *Pi2*, forms the P-layer of the cipher. The *KeySchedule* is very simple – a so-called working key is derived from the secret key and then XORed to the state at each round. For the encryption procedure, the working key is simply equal to the secret key. Note that the self-inverse property of the cipher has big advantages when both encryption and decryption need to be implemented on the same circuit.

MINI-AES [12], or a small scale variant of AES, has been described by Cid, Murphy, and Robshaw in 2005 in order to provide a suitable framework for comparing different cryptanalytic methods. In this paper, we consider a 10-round MINI-AES with a block-size of 64 bits, a key length of 64 bits, and a word size of 4 bits. The main difference between AES and the version of MINI-AES we chose to examine is that the S-box and the MDS matrix are defined over the field GF(2^4). Therefore, the selected instance of MINI-AES can be considered as a lightweight version of the AES cipher.

MCRYPTON [29] is a 64-bit block cipher supporting three different key length (64, 96, and 128 bits), designed by Lim and Korkishko in 2006 and is one of the first lightweight SPN block ciphers. Each round of MCRYPTON consists of the following operations: *NonLinear Substitution* γ, *Column-wise bit Permutation* π, *Column-to-row Transposition* τ, and *Key Addition* σ. The operation γ (S-layer) consists of 16 nibble-wise substitutions using four 4-bit S-boxes (S_0, S_1, S_2, S_3, all affine equivalents to the inversion in GF(2^4) and such that $S_2 = S_0^{-1}$ and $S_3 = S_1^{-1}$). The P-layer consists of π and τ operations. The π operation is an involutional bit-wise matrix multiplication. The τ operation simply transposes the state and is thus an involution. The *KeySchedule* is simple and consists of two stages: a round key generation through a nonlinear S-box transformation and a key variable update through a simple rotation. Finally, the σ operation XORs the round key to the state. Independent of the key length, MCRYPTON always uses 12 rounds with a slightly different *KeySchedule*. Note that decryption and encryption can share most of the round operations and that the *KeySchedule* allows a direct derivation of the last round key.

PRESENT [9], designed by Bogdanov et al., was proposed in 2007 and established itself as one of the most prominent lightweight block ciphers. It has recently been adopted as a standard in ISO/IEC 29192-2. The 31-round cipher has a block-size of 64 bits and comes with an 80-bit or 128-bit key. Each round of PRESENT consists of the following operations: *sBoxLayer*, *pLayer* and *AddRoundKey*. The *sBoxLayer* is defined as the simultaneous application of a very light 4-bit S-Box to each nibble of the state. The *pLayer* is a simple bitwise permutation. The *KeySchedule* rotates the key variable, XORs a constant and applies the S-box to the key variable. *AddRoundKey* XORs the 64 most significant bits of the key variable to the state. Note that the *pLayer* provides a rather slow diffusion of the cipher, which results in the considerably high number of rounds.

KLEIN [18] is a rather young lightweight cipher proposed by Gong, Nikova, and Law in 2010. It is a block cipher with a fixed 64-bit block-size and a variable key length of 64, 80 or 96 bits. Each round of the cipher consists of the

following operations: *SubNibbles*, *RotateNibbles*, *MixNibbles*, and *AddRoundKey*. The operation *SubNibbles* (S-layer) is defined as the simultaneous application of an involutive 4-bit S-Box to each element of the state. The P-layer consists of *RotateNibbles* and *MixNibbles* operations. The *RotateNibbles* operation rotates the state two bytes to the left. The *MixNibbles* coincides with the AES *MixColumns* operation, i.e. pre-multiplies each column of the state by an MDS matrix defined over GF(2^8). The *KeySchedule* derives the round key from the secret key, by applying two S-Boxes and some simple linear operations. Finally, the *AddRoundKey* XORs the round key to the state. KLEIN-64/80/96 uses 12/16/20 rounds respectively.

LED [20], designed by Guo, Peyrin, Poschmann, and Robshaw in 2011, is one of the most recent lightweight ciphers. It is a nibble-based 64-bit block cipher with two variants taking 64-bit and 128-bit keys. Each round of LED consists of the following operations: *AddConstants*, *SubCells*, *ShiftRows*, and *MixColumnsSerial*. Once every 4 rounds the *AddRoundKey* operation is applied. The *SubCells* (S-layer) reuses the PRESENT S-box and applies it to each 4-bit element of the state. *MixColumnsSerial* uses an MDS matrix defined over GF(2^4) for linear diffusion that is suitable for compact serial implementation since it can be represented as a power of a very simple binary matrix. *AddConstants* XORs a constant to the state at each round. *ShiftRows* operates by rotating row i of the array state by i cell positions to the left. *AddConstants*, *ShiftRows*, and *MixColumnsSerial* form the P-layer of the cipher. The 64-bit key variant consists of 32 rounds while the 128-bit key variant consists of 48 rounds. The cipher has no *KeySchedule*, meaning the same key is XORed to the state using *AddRoundKey*, once every 4 rounds.

The resulting set of block ciphers represents a wide spectrum of building blocks for the S-layer, P-layer, and the key schedule. In summary, AES is (the only) byte-oriented block cipher (i.e. byte-based S- and P-layers) with an MDS P-layer; NOEKEON has a nibble-based S-layer, a bit-based P-layer and it is a self-inverse bit-sliced cipher; MINI-AES is a nibble-oriented cipher (i.e. nibble-based S- and P-layers) with an MDS P-layer; MCRYPTON has a nibble-based S-layer, a bit-wise matrix for the P-layer with a specific key schedule; PRESENT has a nibble-based S-layer and a very simple bit permutation for the P-layer; KLEIN has a nibble-based S-layer and a byte-based MDS P-layer (equivalent to AES); finally, LED is a nibble-oriented block cipher (i.e. nibble-based S- and P-layers) with an MDS P-layer and no key schedule.

Note that KLEIN and AES share the same MDS matrix; LED and PRESENT share the same S-layer; MINI-AES and LED have different nibble oriented MDS matrices; and the S-layer of MINI-AES and MCRYPTON are close (affine equivalents) to each other. Therefore, we have a variety of building blocks: bit-, nibble- and byte-oriented blocks; different complexity of S-boxes; either simple matrices, the MDS ones, or just a simple permutation as the P-layer. All this allows us to investigate how different elements influence the overall performance when low-latency encryption is the ultimate goal.

3 Hardware Evaluation

In this section, we provide an extensive hardware evaluation of the seven block ciphers which we identified in the previous section. Besides the cryptographic properties of a cipher, the chosen architecture has a significant influence on the overall performance. As our goal is to evaluate designs with the lowest achievable latency, we mainly focus on 1-cycle and 2-cycle based architectures. More specifically, a 1-cycle based architecture represents a fully-unrolled architecture which requires a single clock cycle for its execution. Similarly, a 2-cycle based architecture needs two clock cycles in order to execute its computation. Since the term low-latency implies a low number of clock cycles for the algorithm execution (recall the systems with a limited clock frequency), we do not evaluate architectures that require three or more clock cycles.

We then distinguish between encryption (ENC) only and encryption/decryption (ENC/DEC) architectures. Moreover, as will become apparent later, some of the implemented ciphers benefit from the inherent similarities between encryption and decryption datapaths. In these cases, we also provide figures for a more compact but still slightly slower implementation that shares the datapath. Figure 2 depicts all the evaluated architectures, however for readability we only report results for (ENC/DEC) architectures. The results for the architectures supporting encryption only are provided in Appendix B.

The presented results are obtained in 90 nm CMOS technology, synthesized with the Cadence RTL compiler version 10.10-p104. In order to have a better overview on the hardware performance, we always provide figures for both time-constrained and unconstrained designs. By time-constrained, we mean a design that achieves the minimum possible critical path at the expense of a large area overhead. An unconstrained design consumes the minimum possible area with the drawback of being a slower circuit. In both cases, this only refers to the synthesis tool constraints and not to the actual RTL code, which in fact remains

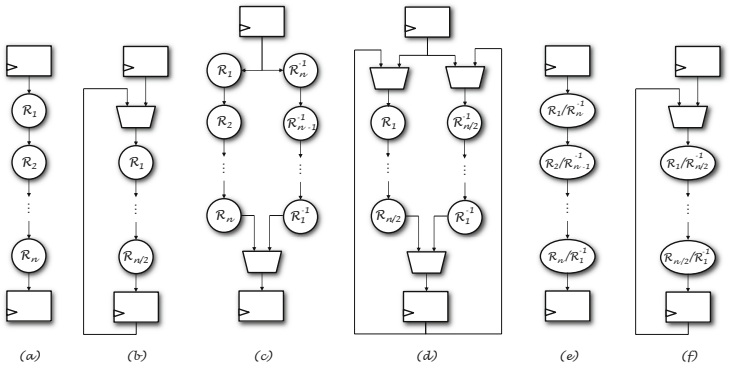

Fig. 2. Six evaluated architectures: (a) 1-cycle based, ENC-only. (b) 2-cycle based, ENC-only. (c) 1-cycle based, ENC/DEC. (d) 2-cycle based, ENC/DEC. (e) 1-cycle based, ENC/DEC, shared datapath. (f) 2-cycle based, ENC/DEC, shared datapath.

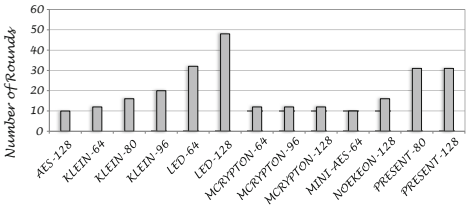

Fig. 3. Number of rounds of the tested ciphers

the same. The code of all designs is written in Verilog and tested against the available test vectors. The data showing the implementation results is provided in Table 1, Appendix A.

Although we rank the ciphers according to their hardware performance, we do not attempt to define *the most efficient* one with respect to *all* evaluated criteria. We believe that depending on the application requirements, the selection of the most efficient design could be based on any of the following criteria: area, latency, time-area product, power, or energy. Moreover, if more than one criterion influences the final decision, we believe that it is rather trivial to combine the presented data and obtain a unique benchmark. As the evaluated ciphers provide different security levels, there is no easy way to fairly compare them against each other. While it is rather obvious that a cipher with a block-size of 64 bits will perform better in terms of area than one with 128 bits, the influence of the key length remains rather vague. With this evaluation, we bring to light the influence of similar and other design decisions on the final hardware performance.

Recall that all the evaluated ciphers have a block-size of 64 bits, except AES and NOEKEON which have a 128-bit block-size. Some ciphers support different key lengths and therefore we evaluate 128-bit key AES; 64-bit, 80-bit, and 96-bit key KLEIN; 64-bit and 128-bit LED; 64-bit, 96-bit, and 128-bit key MCRYPTON; 64-bit key MINI-AES; 128-bit key NOEKEON; and 80-bit and 128-bit key PRESENT. Finally, as most of the obtained results will be highly correlated with the number of cipher rounds, we provide Fig. 3, which visualizes this metric.

3.1 Latency

We define latency as a measure of time needed for a certain design to complete a defined (computational) task. In our context, the computational task is defined as an encryption of a single message block and the latency is calculated as:

$$\text{Latency} = N \cdot t_{cp} \ ,$$

where N is the number of clock cycles needed for the encryption of a single message block and t_{cp} is the critical path of the circuit. In order to highlight the difference between latency and throughput, we outline that the latency truly depends on the inherent properties of a cryptographic algorithm, while the throughput does not – it can be simply increased using the common signal processing techniques such as pipelining and parallel computations.

Figure 4 shows the minimum achievable latency for the ENC/DEC module of all the evaluated ciphers. NOEKEONs-128 denotes a NOEKEON implementation with a shared datapath for encryption and decryption, and it is clearly marked in gray to set it apart from the other designs. The figure further reveals that, in general, there is only a slight advantage of 1-cycle based architectures over 2-cycle based ones, but minimal latency is obtained with a 1-cycle based architecture as expected. The designs that show the highest performance are certainly MINI-AES and MCRYPTON (all key lengths). Being around 30 % slower, KLEIN-64 is the third best candidate. The lowest performance comes from LED-128, which is more than 5 times slower than MINI-AES. AES, for example, achieves 70 % slower critical path than MINI-AES.

What is interesting to observe is that the latency of certain designs, i.e. KLEIN and LED, depends on the key length, while for others, i.e. MCRYPTON and PRESENT, this is not the case. This links directly to the number of rounds, which in case of KLEIN and LED increases for larger key lengths, while it remains constant for MCRYPTON and PRESENT (recall Fig. 3).

In Appendix B, we provide the results for ENC-only architectures (see Fig. 12) The results show that the performance of certain designs, e.g. KLEIN-64 and MINI-AES, certainly degrades when the decryption path is embedded into the design. In order to explain this in more detail, we provide Fig. 5 where we depict the average latency per round of each cipher for both ENC-only and ENC/DEC architectures. It is easy to see that the decryption datapath of AES, KLEIN, and MINI-AES is considerably slower than that of the encryption. To a lesser extent this also holds for LED, MCRYPTON, and PRESENT. NOEKEON is the only cipher that does not suffer from this property. We also observe a clear correlation between the average latency per round and the complexity of the round.

An unconstrained design of PRESENT, on the other hand, shows a somewhat unexpected result. Its unconstrained ENC-only architecture (see Fig. 12, Appendix B) seems to be slower than its ENC/DEC architecture. This result is explained by the fact that, when unconstrained, the synthesis tool optimizes designs for area while the timing is less important. When time-constrained, however, the synthesis tool makes a significant effort to optimize for timing and therefore the ENC/DEC architecture of PRESENT becomes slower than its ENC-only architecture.

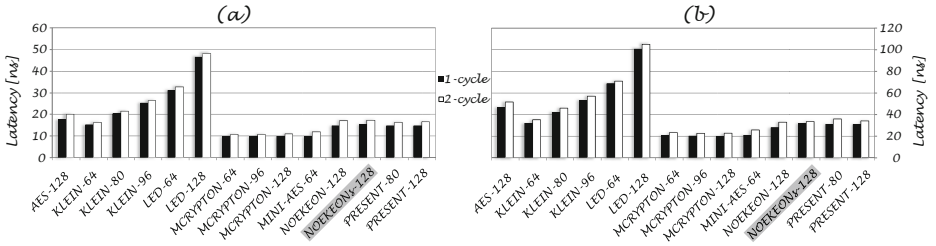

Fig. 4. Minimum latency [ns] for ENC/DEC module: (a) Time-constrained. (b) Unconstrained.

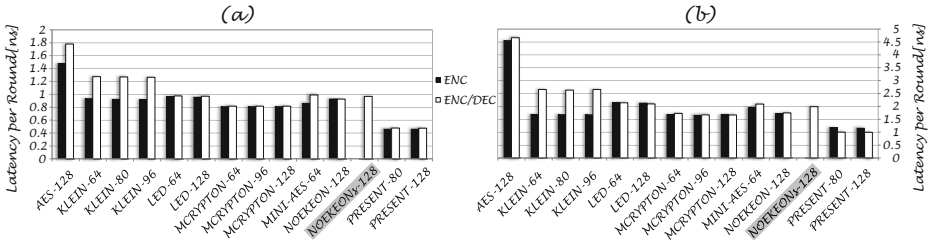

Fig. 5. Average latency [ns] per round: (a) Time-constrained. (b) Unconstrained.

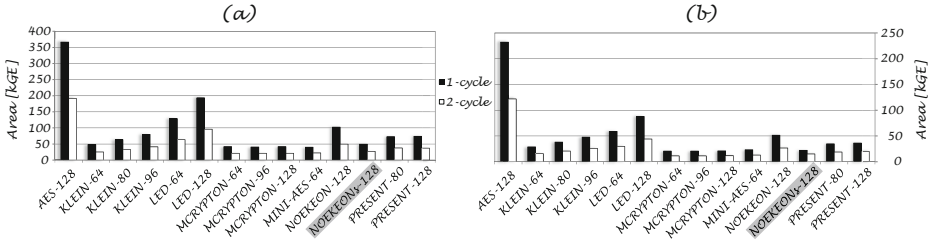

Fig. 6. Minimum area [kGE] for ENC/DEC module: (a) Time-constrained. (b) Unconstrained.

Finally, we note that the ratio between the latency of the unconstrained and time-constrained designs ranges from 2.63 for AES (ENC/DEC) to only 1.30 for NOEKEON (ENC-only, Fig. 12, Appendix B), which illustrates the *elasticity* of the design's latency.

3.2 Area

Similar to the previous subsection, we first provide results for the circuit size of all the evaluated cipher variants. Secondly, we elaborate on the area per round distribution, where we observe several interesting results. Note that the area is expressed in gate equivalence (GE) units, representing the relative size of the circuit compared to a simple 2-input NAND gate.

Figure 6 illustrates the area for ENC/DEC architectures. In contrast to the latency figures, the advantage for 2-cycle based architectures is clear: 2-cycle based architectures consume approximately half of the area of the 1-cycle based architectures. We also observe a significant correlation between the number of cipher rounds and the circuit size. MINI-AES and MCRYPTON again show the best result, followed by the approximately 25 % larger KLEIN-64 implementation. PRESENT comes as the next one with about 60 % overhead. Not surprisingly, the largest circuit size is shown by AES, which is more than 9 times larger than MINI-AES. From the lightweight ciphers, LED-128 consumes the biggest area and it is more than 4 times larger than MINI-AES.

An interesting property can be observed in NOEKEON where due to their inherent similarity the datapaths for encryption and decryption can be shared.

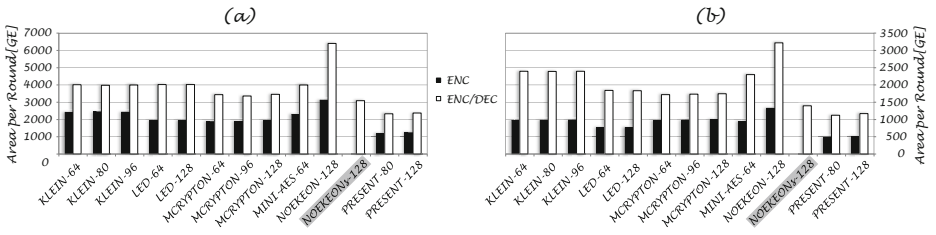

Fig. 7. Average area [GE] per round: (a) Time-constrained. (b) Unconstrained.

Denoted with NOEKEONs-128 (grayed) in Fig. 6 it can be seen that an implementation with shared datapath results in significant area savings (about 50 %), while not influencing the latency as much, i.e. only about 5 % increase (recall Fig. 4). A similar observation, still to a lesser degree, is true for MCRYPTON. When implemented with a shared datapath (not depicted) this results in about 30 % area savings with about 20 % timing overhead compared to the results depicted in Fig. 4. Although encryption and decryption look quite similar for MCRYPTON, two layers of multiplexors per round are needed in the shared datapath in order to choose the correct S-boxes. This extra logic multiplied by the unrolling factor results in quite a significant latency and area overhead for the total design.

Figure 7, which illustrates the average area per round for each cipher (except AES, since its round size goes well beyond the other values – 23 kGE for unconstrained and 37 kGE for time-constrained), shows that PRESENT has the smallest round amongst all ciphers, which is not surprising, as its round consists of an S-layer and a very light P-layer (wiring only). The P-layers of other ciphers involve more complex operations such as multiplication with an MDS matrix for MINI-AES, for example, or variations thereof for other ciphers. Note also that the average area per round of NOEKEON is relatively large. This is due to its block size of 128 bits; twice that of the other ciphers. This only confirms our initial assumption that both the number of cipher rounds and their complexity have a significant influence on hardware performance.

There are a number of observations about the area per round distribution that we illustrate here using KLEIN-80 as an example (see Fig. 8); although the same observation holds to a higher or lesser extent for most of the evaluated ciphers. The first is that due to the higher complexity of decryption, the critical path passes through the decryption datapath, which therefore becomes considerably larger than the encryption datapath when time-constrained. NOEKEON is the only cipher exempt from this effect, while the effect is barely noticeable in the case of LED. When constraints are relaxed, this effect naturally fades away, although remaining slightly noticeable even in unconstrained implementations.

Another observation that can be made for both time-constrained and unconstrained implementations, and holds over all the evaluated ciphers, is the considerably smaller area taken by the last few rounds of an unrolled design. For example in the time-constrained implementation of KLEIN-80, the last round is more than

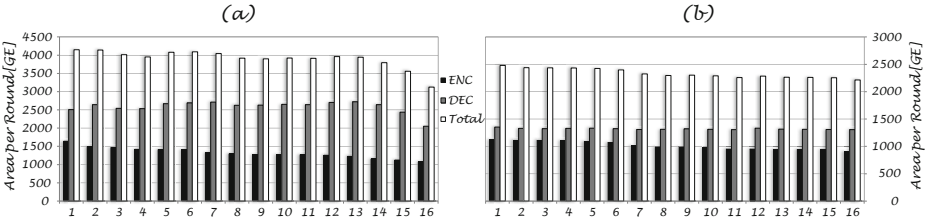

Fig. 8. Area distribution [GE] per round of KLEIN-80: (a) Time-constrained. (b) Unconstrained.

25 % smaller in size than the largest (in this case the second) round. For all other ciphers this difference always remains above 20 %. This phenomenon is explained by the fact that the logic gates used in the last rounds require considerably lower driving strength since they drive less logic than the middle rounds and can therefore be smaller. We further address this observation in Section 4.

The third observation that could be drawn from Fig. 8 is a noticeable swing in area in the first 13 rounds of the time-constrained KLEIN-80 implementation (similar observation holds for all other ciphers as well). This is however an effect introduced by the synthesis tool and is caused by insertion of a significant number of buffer cells in order to strengthen (and thus speed up) the signal propagation throughout the combinational network of the circuit which happens periodically, several rounds after each other.

The ratio between the size of time-constrained and unconstrained designs spans the range from 1.66 for KLEIN-80 to 2.22 for NOEKEONs. This ratio defines the *elasticity* of the design's area and is an indication of the overhead in area needed to achieve the smallest possible critical path of the design.

3.3 Time-Area Product

Although it is a simple combination of the two previously described metrics, we still provide graphs for the time-area product as this is an often used criterion for selecting the final implementation. Figure 9 illustrates the time-area product for the ENC/DEC architecture.

Again, the highest performance with respect to this criterion is shown by MINI-AES and all the versions of MCRYPTON. With more than 60 % overhead, KLEIN-64 takes the third place, while the lowest performance is again shown by LED-128. For all the tested ciphers it holds that the 2-cycle based architecture provides between 40 % and 45 % more efficiency with respect to this metric.

When moving from unconstrained to time-constrained designs the highest gain is shown by AES with 40 % decrease of the time-area product, while NOEKEONs achieves even a negative gain with 8 % increase of the time-area product. In general this ratio (time-area of unconstrained versus time-area of time-constrained designs) ranges between 0.85 and 1.00 which reflects in a rather small overall improvement.

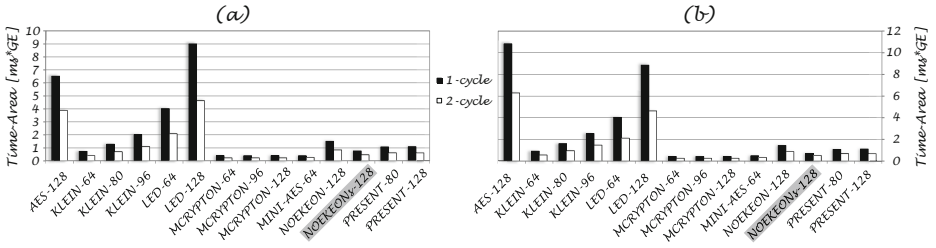

Fig. 9. Minimum time-area product [ms·GE] for ENC/DEC module: (a) Time-constrained. (b) Unconstrained.

3.4 Power and Energy

The results for the average power consumption are obtained by taking into account the switching activity of the circuit and are based on synthesis results. While accurate power measurement is only possible once the circuit is manufactured, we believe that our estimates are still reliable when it comes to comparing the power consumption between different designs. We note here that the term *average* is relative, since we consider designs with very low latency. Therefore, when considering a fully unrolled design (1-cycle), the average power is measured, and hence averaged, over a single clock cycle which in fact reflects the instantaneous power consumption. For the 2-cycle based designs, the power is averaged over two clock cycles. In order to eliminate the data dependency, we average the power consumption over 100 random vector inputs for each measurement.

Since the power consumption is linearly related to the operating frequency, this metric directly influences the value of the measured power. Our strategy of setting the operating frequency is simple in this case – we set the frequency as the reciprocal of the critical path. Therefore, the power consumption of each design is measured during its shortest possible execution time. The energy consumption is normalized over the number of processed bits, i.e. the message block-size, and calculated as:

$$E = \frac{P \cdot \text{Latency}}{B} = \frac{P \cdot N \cdot t_{cp}}{B} \ ,$$

where P is the average power, N is the number of clock cycles needed for the encryption of a single message block, t_{cp} is the critical path of the circuit, and B is the message block-size.

Figures 10 – 11 illustrate the power and energy consumption, respectively. The most power and energy efficient designs are again MINI-AES, MCRYPTON, and KLEIN-64, while LED consumes the most. Surprisingly, a large design such as AES consumes much less energy than most of the lightweight ciphers. This in fact relates to the number of rounds, which in case of AES is only 10, as well as to its block size of 128 bits (energy is normalized over the block size).

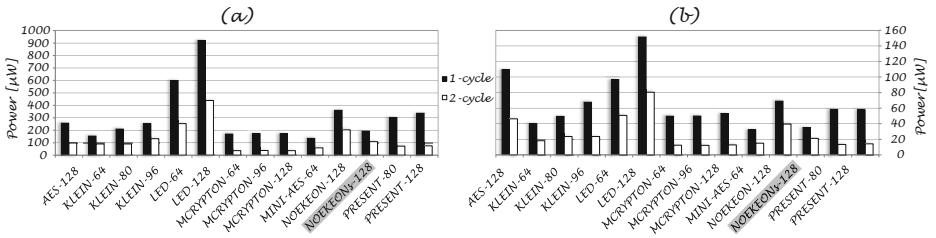

Fig. 10. Power consumption [μW] for ENC/DEC module: (a) Time-constrained. (b) Unconstrained.

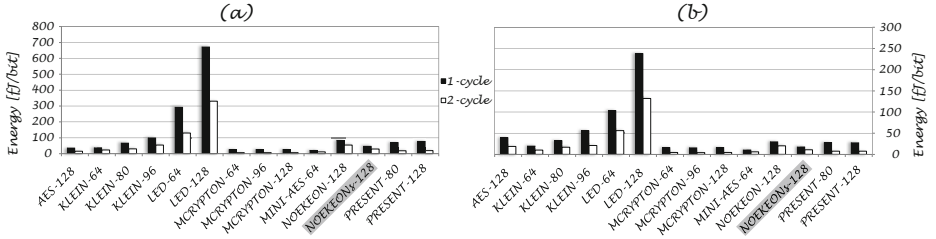

Fig. 11. Energy consumption [fJ/bit] for ENC/DEC module: (a) Time-onstrained. (b) Unconstrained.

4 Discussion and Conclusions

The ciphers we have evaluated within our framework are mainly designed for lightweight applications. They were not designed to satisfy the low-latency requirement imposed by new applications. Therefore, some of the ciphers which provide very good lightweight properties, e.g. LED and PRESENT, demonstrate quite a low hardware performance when it comes to the low-latency behavior. Still, we believe that by looking at the solutions offered by lightweight cryptography and understanding how their inherent properties influence the low-latency behavior one makes the very first step towards building an efficient low-latency cryptographic primitive. We summarize our results and give some guidelines for designing low-latency algorithms. In this context, we mainly address hardware properties of the algorithms.

S-Box. AES is the only cipher with an 8-bit S-box which is significantly larger than the 4-bit S-boxes used by the other ciphers. In theory, a cryptographically strong 8-bit S-box is on average 32 times larger than a cryptographically strong 4-bit S-box. In practice, due to the characteristics of standard cell libraries, this ratio is smaller but remains around 20. This fact strongly encourages the use of cryptographically strong 4-bit (or even 3-bit) S-boxes where possible. We stress here that even among the 4-bit (or 3-bit) S-boxes there are significant differences in circuit size [28].

Number of Rounds. Although both LED and PRESENT use 4-bit S-boxes, thus having a relatively lightweight round, the number of rounds they consist of is

considerably large (see Fig. 3). When a design is (partially) unrolled, the number of rounds becomes a significant factor in the algorithm's performance. While this is obvious in the context of the circuit's latency, once we target low-latency design, also the area overhead becomes significant. This implies a higher power and energy consumption as well. We therefore suggest to minimize the number of rounds of the cryptographic algorithm.

Round Complexity. An interesting conclusion comes from comparing for example the MINI-AES and PRESENT algorithms. While the PRESENT round is very lightweight (it consist of the S-layer and the P-layer, which is in fact only wiring in hardware), the algorithm still needs a relatively large number of rounds in order to achieve good cryptographic properties. MINI-AES, on the other hand, has only 10 rounds and achieves good cryptographic properties by having a heavier P-layer, i.e. an MDS matrix, which efficiently increases the number of active S-boxes at low-cost. To illustrate, the P-layer of MINI-AES is about 30 % larger than its S-layer and therefore 10 rounds of MINI-AES versus 31 rounds of PRESENT seem to be a very good design choice. We, therefore, suggest to reduce the number of rounds at the cost of (slightly) heavier round. Finding a lightweight P-layer with good cryptographic properties is of a high importance here. Similar to MINI-AES, MCRYPTON demonstrates a very good selection for the P-layer (a bitwise matrix multiplication) while KLEIN's P-layer (a byte oriented MDS) seems to be rather heavy.

Key Schedule. When comparing KLEIN and LED on one side with MCRYPTON and PRESENT on the other, we observe that the number of rounds of KLEIN and LED increases with the key length, which is certainly an undesired property. This is not the case with MCRYPTON and PRESENT where the number of rounds remains constant even if the key length changes. Additionally, LED and NOEKEON ciphers come without key schedule, i.e. the same round key is used in all rounds. Although the key schedule is not within the critical path, this feature reduces the complexity of the circuit and it is, therefore, beneficial for the implementation cost of low-latency designs.

Heterogenous Constructions. As we already observed in Fig. 8, the last few rounds of the unrolled implementations are smaller in area than the middle ones. This leads to an interesting conclusion: we suggest to design cryptographic primitives with heterogenous rounds. Namely, designing the algorithm such that the last few rounds are more complex, and thus larger in area, would reduce the number of rounds and reduce the complexity of the whole design. This would, obviously, have consequences for lightweight (round-based) implementation of the algorithm, but here we only consider the low-latency requirements. To further illustrate this observation, we provide Fig. 17 in Appendix C, where the area per round distribution is given for the PRESENT-80 block cipher assuming several different timing constraints.

Encryption and Decryption Procedures. Although Fig. 8 shows only the results for KLEIN, it illustrates a trend common to all ciphers (except NOEKEON). The figure clearly shows that there is a noticeable imbalance between the

encryption and decryption datapaths for most of the tested ciphers. The explanation of this phenomenon is rather simple. Most of the ciphers are designed with the efficiency of the encryption procedure in mind. Therefore, the S-box and the P-layer are often chosen such that their complexity is smaller than that of their inverses. This fact indeed favors the approach of NOEKEON, where the same hardware resources can be reused for both encryption and decryption. This approach not only saves a significant amount of area, but also reduces the latency of the implementation. We also observe that although MCRYPTON has (nearly) involutional layers there is a non-negligible cost to reuse them for both encryption an decryption (due to the required insertion of multiplexors).

Conclusion. We have introduced the domain of low-latency encryption, clearly distinguishing it from the domains of lightweight and conventional encryption. Six well-known lightweight SPN block ciphers, including AES, were selected based on their properties and identified as possible candidates to yield good low-latency behavior. We evaluated their hardware performance within the context of low-latency encryption, thereby providing the first results in the field. It has been shown that the obtained results (i.e. latency, area, power, and energy consumption) are strongly influenced by the design properties such as the number of rounds, the round's complexity, and the similarity between encryption and decryption procedures. We hope that our results will inspire others to design new and efficient low-latency cryptographic primitives.

References

1. FIPS Pub. 197: Specification for the AES (November 2001), `http://csrc.nist.gov/pub-lications/fips/fips197/fips-197.pdf`
2. Biryukov, A. (ed.): FSE 2007. LNCS, vol. 4593. Springer, Heidelberg (2007)
3. Robshaw, M., Billet, O. (eds.): New Stream Cipher Designs. LNCS, vol. 4986. Springer, Heidelberg (2008)
4. Mangard, S., Standaert, F.-X. (eds.): CHES 2010. LNCS, vol. 6225. Springer, Heidelberg (2010)
5. Preneel, B., Takagi, T. (eds.): CHES 2011. LNCS, vol. 6917. Springer, Heidelberg (2011)
6. Aumasson, J.-P., Henzenz, L., Meier, W., Naya-Plasencia, M.: Quark: a lightweight hash. In: Cryptographic Hardware and Embedded Systems — CHES 2010 [4], pp. 1–15
7. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Keccak sponge function family main document (version 2.1). Submission to NIST (2010), `http://keccak.noekeon.org/Keccak-main-2.1.pdf`
8. Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varici, K., Verbauwhede, I.: SPONGENT: A lightweight hash function. In: Cryptographic Hardware and Embedded Systems — CHES 2011 [5], pp. 312–325
9. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)

10. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
11. Cannière, C.D., Preneel, B.: Trivium. In: The eSTREAM Finalists [3], pp. 244–266
12. Cid, C., Murphy, S., Robshaw, M.J.B.: Small Scale Variants of the AES. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 145–162. Springer, Heidelberg (2005)
13. Daemen, J., Peeters, M., Rijmen, V., Assehe, G.V.: Nessie Proposal: Noekeon (2000), `http://gro.noekeon.org/`
14. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
15. European Network of Excellence in Cryptology – ECRYPT. The eSTREAM Project (2004), `http://www.ecrypt.eu.org/stream/`
16. Feldhofer, M., Rechberger, C.: A Case Against Currently Used Hash Functions in RFID Protocols. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops, Part I. LNCS, vol. 4277, pp. 372–381. Springer, Heidelberg (2006)
17. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES Implementation on a Grain of Sand. IEE Proceedings Information Security 152(1), 13–20 (2005)
18. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: A New Family of Lightweight Block Ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
19. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 222–239. Springer, Heidelberg (2011)
20. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED Block Cipher. In: Cryptographic Hardware and Embedded Systems — CHES 2011 [5], pp. 326–341
21. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain Family of Stream Ciphers. In: The eSTREAM Finalists [3], pp. 179–190
22. Hodjat, A., Verbauwhede, I.: Area-Throughput Trade-offs for Fully Pipelined 30 to 70 Gbits/s AES Processors. IEEE Transactions on Computers 55(4), 366–372 (2006)
23. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
24. Izadi, M., Sadeghiyan, B., Sadeghian, S., Khanooki, H.: MIBS: A New Lightweight Block Cipher. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 334–348. Springer, Heidelberg (2009)
25. Kavun, E., Yalcin, T.: A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 258–269. Springer, Heidelberg (2010)
26. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.: PRINTcipher: A Block Cipher for IC-Printing. In: Cryptographic Hardware and Embedded Systems — CHES 2010 [4], pp. 16–32
27. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: 14th International Workshop on Fast Software Encryption — FSE 2007 [2], pp. 196–210
28. Leander, G., Poschmann, A.: On the Classification of 4 Bit S-Boxes. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 159–176. Springer, Heidelberg (2007)

29. Lim, C., Korkishko, T.: mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 243–258. Springer, Heidelberg (2006)
30. Mathew, S., Sheikh, F., Kounavis, M., Gueron, S., Agarwal, A., Hsu, S., Kaul, H., Anders, M., Krishnamurthy, R.: 53 Gbps Native $GF(2^4)^2$ Composite-Field AES-Encrypt/Decrypt Accelerator for Content-Protection in 45 nm High-Performance Microprocessors. IEEE Journal of Solid-State Circuits 46(4), 767–776 (2011)
31. National Institute of Standards and Technology (NIST). Cryptographic Hash Algorithm Competition, `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`
32. National Institute of Standards and Technology (NIST). FIPS 197: Advanced Encryption Standard (November 2001)
33. Poschmann, A., Moradi, A., Khoo, K., Lim, C.-W., Wang, H., Ling, S.: Side-Channel Resistant Crypto for Less than 2,300 GE. Journal of Cryptology 24, 322–345 (2011)
34. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In: Cryptographic Hardware and Embedded Systems — CHES 2011 [5], pp. 342–357
35. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA. In: 14th International Workshop on Fast Software Encryption — FSE 2007 [2], pp. 181–195
36. Standaert, F.-X., Piret, G., Gershenfeld, N., Quisquater, J.-J.: SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 222–236. Springer, Heidelberg (2006)
37. Wheeler, D., Needham, R.: TEA, a Tiny Encryption Algorithm. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 363–366. Springer, Heidelberg (1995)

A Hardware Performance (Data)

In Table 1, we summarize hardware figures for all the tested block ciphers. The best (smallest) values in each column are marked in bold. Since all the values are obtained based on synthesis results, we believe that the metrics including area, latency, and time-area product are estimated with a good accuracy. On the other hand, we believe that accurate power and energy estimation can only be done after place and route is performed and, therefore, we do not provide a detailed report on these two metrics.

Table 1. Hardware performance of all the tested ciphers (90 nm CMOS, synthesis results)

	Time-constrained											
	1-cycle						2-cycle					
	ENC			ENC/DEC			ENC			ENC/DEC		
	L	A	T-A	L	A	T-A	L	A	T-A	L	A	T-A
AES-128	14.8	218.1	3.227	17.8	366.6	6.525	16.6	118.1	1.961	20.2	191.8	3.874
KLEIN-64	11.2	29.0	0.325	15.3	48.2	0.737	12.2	14.6	0.179	16.4	24.9	0.409
KLEIN-80	14.8	39.0	0.577	20.3	63.7	1.293	15.8	19.6	0.310	21.4	32.6	0.697
KLEIN-96	18.4	48.6	0.893	25.3	79.9	2.021	19.6	24.5	0.481	26.4	41.3	1.089
LED-64	30.9	62.0	1.917	31.2	128.7	4.014	32.2	32.2	1.038	32.8	63.5	2.081
LED-128	46.0	93.4	4.296	46.6	193.1	8.999	47.4	47.9	2.269	48.2	96.0	4.625
MCRYPTON-64	9.7	**22.5**	0.218	**9.8**	41.3	0.405	**10.4**	**11.7**	**0.124**	**10.8**	**20.9**	**0.225**
MCRYPTON-96	9.7	22.7	0.221	**9.8**	40.4	**0.396**	**10.4**	12.1	0.126	**10.8**	21.1	0.228
MCRYPTON-128	9.7	23.2	0.225	**9.8**	41.4	0.406	**10.4**	12.1	0.125	11.0	21.0	0.231
MINI-AES-64	**8.6**	23.0	**0.198**	9.9	**40.0**	**0.396**	**10.4**	12.5	0.130	12.0	22.0	0.265
NOEKEON-128	14.9	50.0	0.745	14.8	102.5	1.517	16.6	26.1	0.433	17.0	49.6	0.844
NOEKEONS-128	-	-	-	15.5	49.5	0.768	-	-	-	17.4	27.1	0.471
PRESENT-80	14.3	36.9	0.528	14.8	72.3	1.070	16	19.2	0.308	16.4	37.6	0.616
PRESENT-128	14.3	38.1	0.544	14.7	73.8	1.084	16	19.6	0.313	16.6	37.1	0.615
	Unconstrained											
	1-cycle						2-cycle					
	ENC			ENC/DEC			ENC			ENC/DEC		
	L	A	T-A	L	A	T-A	L	A	T-A	L	A	T-A
AES-128	45.5	103.6	4.715	46.6	232.2	10.820	43	62.3	2.677	51.6	122.0	6.293
KLEIN-64	20.4	11.8	0.240	31.9	28.8	0.918	25.2	7.7	0.194	35.2	15.7	0.553
KLEIN-80	26.9	15.7	0.422	42.1	38.2	1.610	32.2	10.1	0.325	46.0	20.7	0.951
KLEIN-96	33.5	19.7	0.659	53.1	47.9	2.544	39.6	12.6	0.500	57.0	25.8	1.470
LED-64	68.8	24.5	1.688	68.5	58.9	4.038	71	14.8	1.053	71.0	29.7	2.109
LED-128	102.5	36.6	3.754	100.6	88.1	8.858	103.2	21.9	2.258	105.0	44.1	4.629
MCRYPTON-64	20.2	11.7	0.235	20.7	**20.6**	0.427	22	**6.6**	0.146	23.4	**11.3**	0.264
MCRYPTON-96	19.9	11.8	0.235	20.1	20.8	**0.418**	**21**	6.8	**0.143**	**22.6**	11.5	**0.259**
MCRYPTON-128	20.2	12.0	0.242	**20.0**	21.0	0.419	21.2	7.0	0.148	22.8	11.6	0.265
MINI-AES-64	**19.6**	**9.4**	**0.184**	20.9	23.0	0.481	21.6	6.7	0.145	25.8	13.0	0.335
NOEKEON-128	27.6	21.3	0.587	27.9	51.6	1.438	32.4	13.8	0.446	33.0	26.6	0.878
NOEKEONS-128	-	-	-	31.8	22.3	0.710	-	-	-	33.6	15.1	0.507
PRESENT-80	36.6	15.0	0.548	31.0	34.8	1.078	33.6	9.2	0.308	36.0	18.9	0.682
PRESENT-128	35.9	15.7	0.564	30.8	36.3	1.117	33.6	9.7	0.327	34.2	20.0	0.685

L – Latency [ns]
A – Area [kGE]
T-A – Time-Area product [ms×GE]

B Hardware Performance for ENC-Only Modules

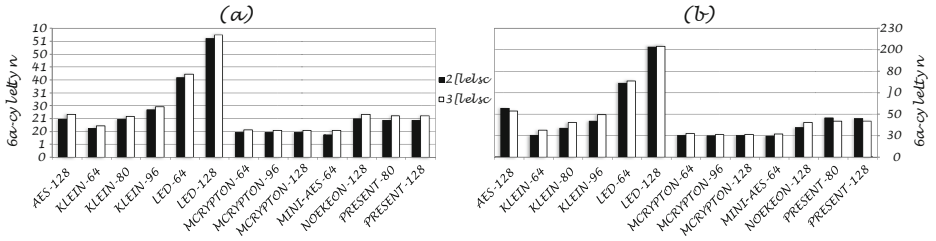

Fig. 12. Minimum latency [ns] for ENC-only module: (a) Time-constrained. (b) Unconstrained.

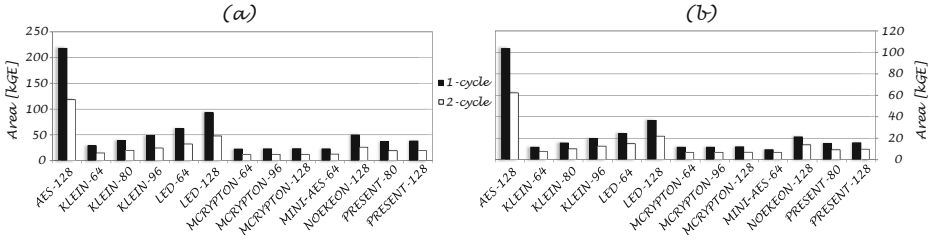

Fig. 13. Minimum area [kGE] for ENC-only module: (a) Time-constrained. (b) Unconstrained.

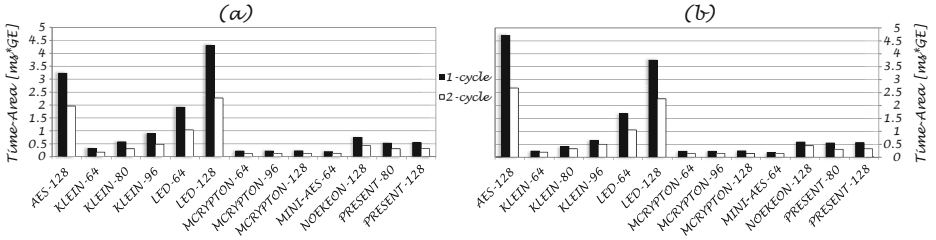

Fig. 14. Minimum time-area product [ms·GE] for ENC-only module: (a) Time-constrained. (b) Unconstrained.

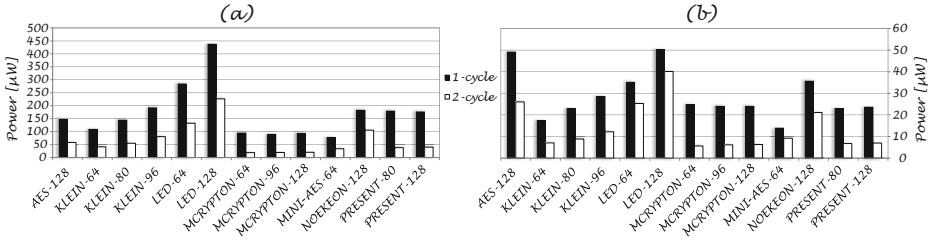

Fig. 15. Power consumption [μW] for ENC-only module: (a) Time-constrained. (b) Unconstrained.

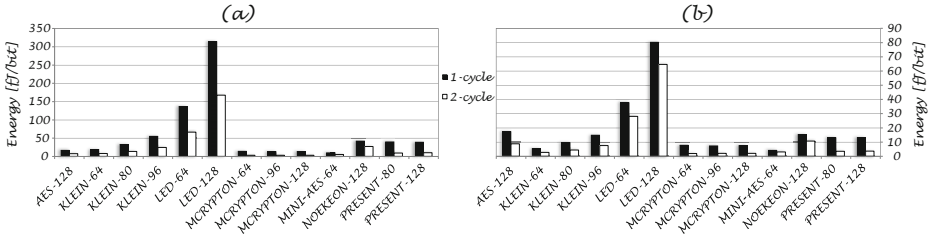

Fig. 16. Energy consumption [fJ/bit] for ENC-only module: (a) Time-constrained. (b) Unconstrained.

C Area per Round Distribution of PRESENT-80 ENC-Only

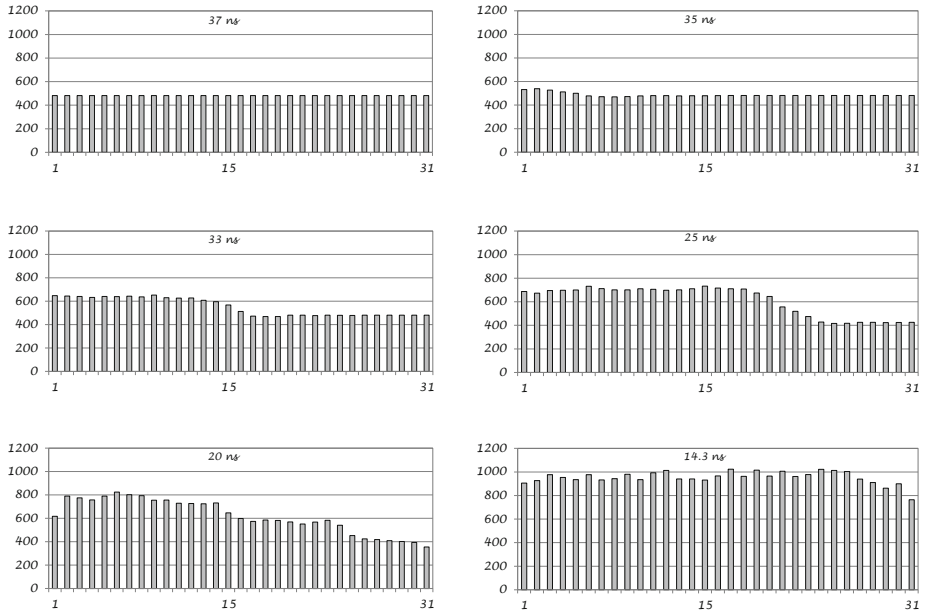

Fig. 17. Area [GE] per round distribution of the PRESENT-80 ENC-only architecture