A Comprehensive Performance Analysis of Hardware Implementations of CAESAR Candidates

Sachin Kumar · Jawad Haj-Yahya · Mustafa Khairallah · Anupam Chattopadhyay

Abstract Authenticated encryption with Associated Data (AEAD) plays a significant role in cryptography because of its ability to provide integrity, confidentiality and authenticity at the same time. Due to the emergence of security at the edge of computing fabric, such as, sensors and smartphone devices, there is a growing need of lightweight AEAD ciphers. Currently, a worldwide contest, titled CAESAR, is being held to decide on a set of AEAD ciphers, which are distinguished by their security, run-time performance, energy-efficiency and low area budget. For accurate evaluation of CAESAR candidates, it is of utmost importance to have independent and thorough optimization for each of the ciphers both for their corresponding hardware and software implementations.

In this paper, we have carried out an evaluation of the optimized hardware implementation of AEAD ciphers selected in CAESAR third round. We specifically focus on manual optimization of the micro-architecture, evaluations for ASIC technology libraries and the effect of CAESAR APIs on the performances. While these has been studied for FPGA platforms and standalone cipher implementation - to the best of our knowledge, this is the first detailed ASIC benchmarking of CAESAR candidates including manual optimization. In this regard, we benchmarked all prior reported designs, including the code generated by high-level synthesis flows.

Detailed optimization studies are reported for `NORX`, `CLOC` and `Deoxys-I`. Our pre-layout results using commercial ASIC technology library and synthesis tools show that optimized `NORX` is 40.81% faster and 18.02% smaller, optimized `CLOC` is 38.30% more energy efficient and 20.65% faster and optimized `Deoxys-I` is 35.16% faster, with respect to the best known results. Similar or better performance results are also achieved for FPGA platforms.

Keywords ASIC Implementation · FPGA · Authenticated Encryption · Logic Optimization · Technology Mapping

S. Kumar, J. Haj-Yahya, A. Chattopadhyay
School of Computer Science and Engineering
Nanyang Technological University
Singapore
E-mail: sachinkumar@ntu.edu.sg,jawad@ntu.edu.sg, anupam@ntu.edu.sg

M. Khairallah
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore
E-mail: mustafam001@e.ntu.edu.sg

1 Introduction

AEAD schemes protect both privacy and authenticity of the message when it is transformed into ciphertext where there may be additional information, such as a packet header, that travels alongside the ciphertext and must get authenticated with it. The need for AEAD emerged from the observation that securely combining a confidentiality mode with an authentication mode could be error prone and difficult [3,23]. Number of practical attacks introduced into production protocols and applications by incorrect implementation, or lack, of authentication (including SSL/TLS)[4].

An AEAD scheme typically consists of two routines. The first one is encryption $\mathcal{E}_K(AD, M)$ which takes as input a shared key K, public associated data AD and the message to be encrypted M and returns a tagged ciphertext C. The second one is decryption/verification $\mathcal{D}_K(AD, C)$, which either returns an invalid symbol $\perp$ if the received ciphertext, associated data and the authentication data do not match, or the decrypted message

M, otherwise. There are three main approaches which are adopted for AEAD: Encrypt then MAC (EtM), Encrypt and MAC (E&M) and MAC then Encrypt (MtE) as shown in Figure 1. In September 2016, the CAESAR competition committee announced the selection of 15 AEAD schemes as candidates for round 3 of the CAESAR competition [10]. This competition signifies the current need for practical, secure and efficient AEAD schemes.

In the survey presented in [1], the round two candidates of the CAESAR competition were categorized into five families on the basis of their base constructions: block cipher-based, stream cipher-based, key-less permutations, hash-function-based and dedicated schemes. AEAD ciphers based on block-cipher allows block-level parallelism while using the underlying block cipher, such as the Offset Code Book mode (OCB) [29,28,24], the Synthetic Counter-in-Tweak mode (SCT) [26] and the Offset Two-Round mode (OTR) [25].

An important aspect of the study of AEAD schemes is the evaluation of their hardware performance, which clearly needs more efforts. So far, nearly all candidates have been supported with a basic hardware implementation [13]. However, the implementations are done on various platforms, for different interfaces. Furthermore, several designs have unique advantages to offer in some platforms, e.g., Field Programmable Gate Array (FPGA). However, FPGA boards are mainly used for verification of the design with the help of programmable gates but it does not provide actual performance metrics of the design which can only be achieved by implementing the design in Application-Specific Integrated Circuits (ASIC). In addition, all the available hardware implementations of the CAESAR competition candidates on the ATHENa hardware evaluation website [13] are fully sequential implementations, i.e. to start processing a new block, all the previous blocks have to be finished. These implementations do not take full advantage of the specific characteristics of the schemes based on the aforementioned modes.

Generally, circuit optimization consists of two phases: logic synthesis and technology mapping. For certain target technologies, such as FPGA, logically optimized circuits do not provide the optimal mapping to the underlying technology, leaving behind a lot of under-utilized hardware resources. This phenomenon is obvious in the AES Sbox circuits proposed by Boyar [8, 9], which are logical optimizations of the circuit proposed by Canright [12]. These circuits are much smaller than the straight-forward ROM-based Sbox in terms of gate count and circuit depth. These two features make them the natural choice for low area ASIC implementations of AES. Interestingly, on the other hand, practical results show that one can achieve a smaller area on FPGA by using the ROM approach [27].

1.1 Contribution

In this paper, hardware evaluation of AEAD ciphers selected in third round of CAESAR competition has been carried out. For ASIC synthesis, we applied delay-optimized and area-constrained synthesis techniques, i.e., in the former case, all AEAD ciphers are kept running until slack is zero in order to achieve maximum speed while in the latter case, all designed are constrained to achieve minimum hardware area at particular frequency. In what follows, benchmarking for both cases have been reported which includes critical path delay (ns), area in Kilo-Gate Equivalent (KGE), throughput (Gbps), throughput per area. Besides that, all AEAD ciphers are compared in terms of throughput per KGE and as well as throughput per energy.

Apart from providing the ASIC benchmarking, detailed optimization studies are reported for NORX, CLOC and Deoxys-I. Our pre-layout results using commercial ASIC technology library and synthesis tools show that optimized NORX is 40.81% faster and 18.02% smaller, optimized CLOC is 38.30% more energy efficient and 20.65% faster and optimized Deoxys-I is 35.16% faster, with respect to the best known results as depicted in Table 1.

In addition, the effect of CAESAR APIs on performance might be significant as the use of API favors iterative implementations. The overhead of API was analyzed and its effect on the overall performance of the ciphers have been discussed in this paper. Moreover, qualitative analysis has been carried out about the effect of technology-mapping and technology aware optimization on the cipher performance. The reason being that some of the optimization fit more for FPGA and other suitable for ASIC, while the current hardware benchmarking in the CAESAR competition uses only one circuit for all technologies.

1.2 Organization

The remaining of this paper is organized as follows. Section 2 is about the related work and motivation which includes introduction of AEAD ciphers, its FPGA bench marking and purpose of this work. Detailed explanation of AEAD ciphers selected in third round of CAESAR competition and their hardware evaluation in ASIC are presented in section 3. In section 4, Optimized NORX, Deoxys, CLOC designs as a case study are reported along with their simulation results. Section 5 provides

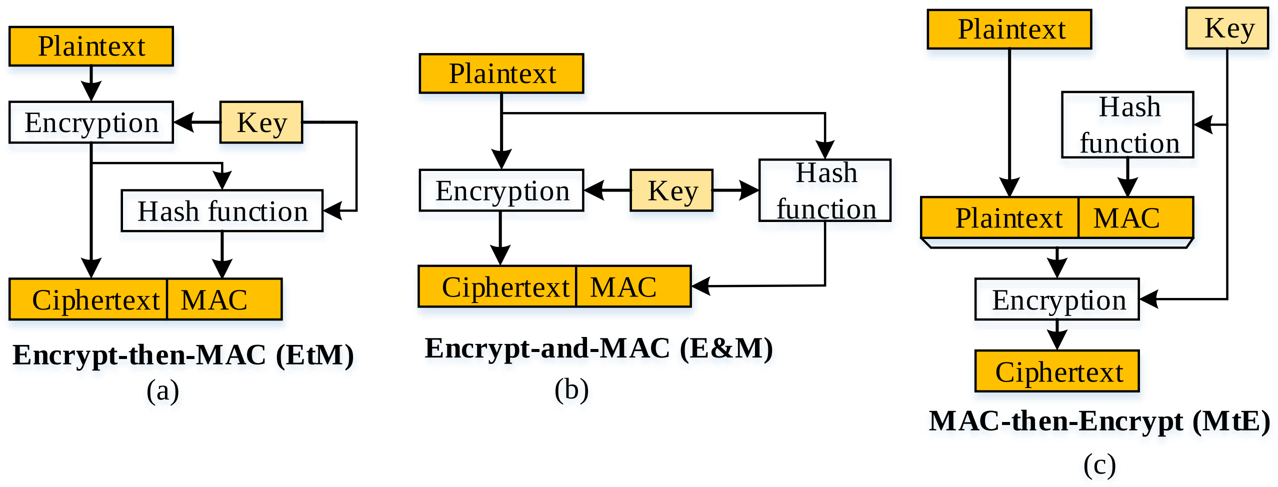

Fig. 1: Approaches to Authenticated Encryption with Associated Data

Table 1: Synthesis results of the `Deoxys-I-128`, `NORX` and `CLOC` implementation using TSMC 65nm technology

Design	Area (KGE)	Max. Freq. (MHz)	Throughput (Mbps)	Efficiency (Mbps/KGE)
Optimized `Deoxys-I` [22]	59.53	847	7,227	121.40
`Deoxys_GMU` [13]	53.37	549	4,684	87.76
Optimized `NORX`	70.13	757.57	83,110	1185
`NORX` [13]	85.54	448.43	57,400	670
Optimized `CLOC`	67.09	746.26	2,850.75	29.50
`CLOC_GMU` [13]	94.87	588.23	3,341.18	24.46

the optimization techniques with respect to technology aware. Section 6 presents the influence of CAESAR API on the performance of the design. At the end, the paper has been summarized in the subsequent section.

2 Related work and motivation

AEAD Ciphers: Authenticated Encryption (AE) is a concept that emerged in the cryptographic community in early 2000s, due to the need for both data privacy and authentication at the same time. In 2000, Charanjit S. Jutla [21] proposed two Encryption modes that also provide message integrity. In 2009, six more modes were standardized by NIST in the ISO/IEC 19772:2009 standard [18]. In 2013, The CAESAR Competition for Authenticated Encryption: Security, Applicability, and Robustness was announced in order to encourage the design of AE algorithms. In round 1, 57 different proposals have been submitted to the competition. Out of these 57 submissions, only 28 submissions qualified to the second round. FPGA implementations of all the 28 candidates have been developed and benchmarked for comparison. In September 2016, 15 candidates have been selected for the third round of the competition. To the best of our knowledge, no studies have been publicly published on the ASIC benchmarking of the candidates.

FPGA Benchmarking: The Cryptographic Engineering Research Group (CERG) at George Mason University (GMU), USA, runs and maintains the online platform ATHENa [13] aimed at fair, comprehensive, and automated evaluation of hardware cryptographic cores targeting FPGAs, Systems on Chip, and ASICs. One of their on-going projects is the comparison of FPGA implementations of the CAESAR competition candidates. They have also provided high-speed round-based implementations of round 2 and 3 candidates. The most recent benchmarking results are published in [16], where the authors provided a summary of available implementations for round 3 candidates that are either designed by the CERG research group or other members of the cryptographic community. The benchmarking process was performed only for FPGA and some of the designs were implemented using High-Level Synthesis (HLS) as opposed to manual Register-Transfer-Level (RTL) design [17]. Moreover, only the implementations that are compliant with the CAESAR Hardware API [15] were considered. The authors of [17] and [16] adopted a few assumptions that motivated their benchmarking process. These assumptions include:

– The rankings of different implementations will be the same regardless of whether the benchmarking is done using FPGA or ASIC.
– The rankings of different implementations will be the same regardless of whether the benchmarking is done using HLS or manual RTL.
– It is only fair to compare implementations with the same hardware API.

Table 2 shows one example of the FPGA benchmarking results targeted for Xilinx Virtex-6 FPGA, ordered descending according to the efficiency (throughput/area).

Motivation: Since the assumptions mentioned earlier are crucial to the benchmarking process, in this paper we discuss their validity. The first challenge to these assumptions was discussed in [22]. The authors in [22] showed that manual algorithm-specific optimization and the use of a customized API can lead to significant gains for these algorithms. Additionally, they showed, using preliminary experimentation, that the rankings from FPGA benchmarking do not extend to ASIC with no change. In fact, ASIC benchmarking should be conducted in a separate process, in addition to FPGA benchmarking. These observations motivated the investigation of the benchmarking process and challenging the previously well-established assumptions. The results of this investigation are discussed in details in sections 4, 5 and 6. However, in the preparation of ASIC benchmarking, we adopted the currently agreed API in order to be consistent with other benchmarking efforts. Nevertheless, we show how the relaxation of API can lead to a very different benchmarking results, which, however, remains consistent to general IC design practices.

3 ASIC Implementation of AEAD ciphers

There are mostly fifteen AEAD ciphers without considering their variants in third round of CAESAR competition [10]. AEGIS is one of them which is constructed by employing `AES` encryption round function except not considering the last round of it. The variants `AEGIS`-128 and `AEGIS`-256 are required 5 `AES` round functions and 6 `AES` round functions respectively in order to process 16 bytes message block. It was claimed by designers that `AEGIS` is faster than `AES` as its computational cost is half of `AES`. Another authentication algorithm submitted to CAESAR known as `AEZ` which is mainly developed with the help of `AES` primitives and enciphering schemes. In this algorithm, the plaintext is appended with a fixed authentication block. Its resulting strings is enciphered with an arbitrary-input-length block cipher which gets tweaked by the nonce and associated data (AD). This process results in strong security and usability properties such as nonce-reuse misuse resistance, user selectable ciphertext expansion. While both `SILC` and `CLOC` are designed by the same authors where `SILC` stands for simple lightweight CFB and `CLOC` stands for compact low-overhead CFB. Both block ciphers are developed for authenticated encryption with associated data (AEAD). It is claimed by the submitters that `CLOC` design is targeted mainly for being secure and optimizing its implementation costs. It can handle short-data input efficiently and is suitable for use with embedded processors. `SILC` is further development of `CLOC` which aims at optimizing the hardware implementation cost of it. It is mainly appropriate for constrained hardware devices. `CLOC` is based on the `AES` block cipher for 16-byte block length and `TWINE` block cipher [30] for 8-byte block length whereas `SILC` is constructed with the help of `AES` block cipher for 16-byte block length and `present` [7] and `LED` [14] for 8-byte block length.

A lightweight authenticated encryption mode called `JAMBU` is based on block cipher `AES`-128 where in `JAMBU` is implemented with the help of bitwise exclusive OR and concatenation operations in order to make it lightweight. Another AEAD cipher, called `COLM`, which is based on `AES`-128 with a key and state of size 128 bits. The aim of `COLM` is to achieve online misuse resistance, to make it fully parallel as well as it should be secure against block wise adaptive adversaries. Another third round candidate, `TIAOXIN`, is nonce-based software oriented authenticated encryption scheme which uses 6 `AES` rounds call per 32-byte messages. In addition, these six `AES` rounds are computed in parallel which leads to be two times faster than `AES`-128 and 3.5 to 6.5 times faster than `AES`-GCM. The authors mentioned that it has been analyzed for various attacks

Table 2: FPGA Benchmarking Results of Round 3 Candidates of the CAESAR Competition

Algorithm	Throughput (Mbps)	Area (LUTs)	Throughput/Area (Mbps/LUT)
MORUS	49,556	3,397	14.5
AEGIS	70,934	3,460	9.3
ACORN	11,304	508	9.1
TIAOXIN	52,796	7,112	7.4
Ketje	24,843	1,238	5.5
NORX	24,519	2,921	5.2
ASCON	5,085	1,270	3.2
AES-OTR	7,708	3,492	1.6
SILC	4,048	3,079	1.3
Keyak	12,600	6,223	1.2
JAMBU_AES	2,008	1,841	1.1
Deoxys	2,882	3,175	0.91
JAMBU-SIMON	939	1,048	0.89
CLOC	2,979	3,143	0.74
OCB	3,109	4,254	0.73
AEZ	3,271	4,730	0.69
COLM	3,109	7,143	0.39

and provides full security nonce-respecting adversaries. The AES-128 based AEAD cipher named as OCB- which stands for "offset codebook" - is a block-cipher mode of operation for efficient authenticated encryption. This cipher is designed to be provably secure and operations performed are in parallel which allows both software and hardware acceleration. Besides this, it has static associated data (AD) feature which means when AD remains unchanged during series of encryption, there is no need to perform operation on AD each time. This results in reducing computational cost when multiple encryption are linked with the same associated data.

A new authenticated encryption design known as Deoxys is based on a tweak-able block cipher Deoxys-BC by employing AES round function as a building block. It has two authenticated encryption modes i.e. Deoxys-I in which nonces are non-repeatable and Deoxys-II in which nonces can be repeated. It has been claimed by the designers that Deoxys provides sufficient protection against linear crypt-analysis in single key model and security against other attacks due to having Deoxys tweakey schedule. Some AEAD ciphers such as ASCON, Ketje and NORX are designed with the help of sponge function and its sister construction-known as duplex construction. In ASCON, there are five states with the size of 64 bits with stronger keyed initialization and keyed finalization function. The algorithm uses two permutation p^a and p^b in which number of rounds a are for the initialization and finalization permutation p^a and b number of rounds are used for the intermediate permutation p^b for processing the associated data and plaintext. Moreover, ASCON round is developed with the help of three operations: constant addition, an nonlinear S-box layer and a linear transformation. In this algorithm, a straightforward key-recovery is prevented by adding extra key addition in the initialization and finalization.

NORX has an unique parallel architecture based on monkey duplex construction [5] along with its two versions i.e. either NORX-32 or NORX-64. This authentication encryption scheme comes with an associated data supporting arbitrary parallelism. In addition, the NORX algorithm is developed with the help of ARX primitives instead of modular addition. This cipher was optimized to be efficient in both software as well as hardware with a single-instruction multiple-data (SIMD)-friendly core and no secret-dependent memory lockups. The underlying permutation f is designed by referencing ChaCha stream cipher where in the integer addition is replaced by a simple bit-wise XOR operation i.e. $(a \oplus b) \oplus (a \wedge b) \ll 1$ which leads to improve its hardware efficiency. Another Monkey Duplex construction based AEAD ciphers are Ketje and Keyak. Ketje has two variants i.e. Ketje_jr and Ketje_sr, these are differentiated in terms of permutation width and size of each. Similarly Keyak is classified into Keyak_lake and Keyak_river which are distinguished by permutation width. Ketje builds on round-reduced versions of Keccak-*f*. The main advantage of Monkey Duplex based ciphers is that it can support different number of calls which eventually helps to invoke the permutation with a different number of rounds. This results in improving the performance of the AEAD schemes by optimizing the number of rounds aggressively.

One more lightweight cipher known as `ACORN` is basically a bit-based sequential authenticated cipher based on linear feedback shift register (LFSR). In `ACORN`, the difference is injected into the state for authentication for better performance. This type of authentication encryption cipher is designed with the help of bit-based sequential stream cipher for the first time. High authentication security is achieved by employing six concatenated linear feedback shift registers. In addition, `ACORN` also allows parallel computation. In `ACORN`, 32 steps can be computed in parallel which benefits high-speed hardware and software implementation. It was claimed by the designer that `ACORN`-128 is more hardware efficient than `TRIVIUM` [11] and `AES`-GCM. One more selected third round candidate, `MORUS` is suitable both for hardware and software efficient implementation. It has three versions; `MORUS`-640-128, `MORUS`-1280-128 and `MORUS`-1280-256 where the first number denotes the state size (in bits) while second number denotes the key size in bits. In `MORUS`, state function is divided into five vectors where size of each vector is either 128-bit or 256-bit depends on the selected variants. There are four operation used in `MORUS`, namely XOR, AND, rotation in words and rotation in subwords. State update function has five similar rounds. Every round modifies two registers; one is modified by rotation in words while other one is with the help of ANDs, XORs and rotation in subwords. In addition, `MORUS` encryption can be seen as stream cipher with a large state which is kept updating continuously in a nonlinear way.

Some of above-mentioned AEAD ciphers (denoted as `cipher_GMU`) are described in VHDL language by ATHENa-Automated Tool for Hardware EvaluatioN while rest are done by the designer themselves. All the above-mentioned AEAD ciphers are functionally verified by Mentor Graphics ModelSim with the help of given test vectors. Each design is synthesized for minimum delay as well as for minimum hardware-area at minimum clock frequency by Synopsys Design Compiler version J-2014.09 using TSMC 65 nm technology standard cell library. In Table 3, lists the throughput(Gbps), critical path delay (ns), area (KGE) and throughput per area etc. parameters of all above-mentioned AEAD ciphers by keeping the constraint for speed-optimization. The calculation of throughput has been done by $\frac{message\ size}{number\ of\ cycles} \times clock\ period$ in which number of cycles have been considered from the ATHENa as to have accurate calculation of throughput.

Since there is always a trade-off between area and delay, for area-optimization process, timing constraint is usually kept unchanged in order to achieve more realistic performance of the design. In what follows, the minimum frequency has been selected from Table 3 which is 448.43 Mhz and then the constraint to achieve minimum hardware area is applied to design compiler tool with the selected frequency. Table 4 lists the hardware area, throughput per area etc. of all the ciphers. With this constraint, it can easily be noticed that synthesis area has been reduced as compared with Table 3. In addition, the trend of hardware area performance at the given frequency of all AEAD ciphers is shown in Fig 3. Apart from listing synthesis results for speed optimization as well as for area optimization in Table 3 and Table 4, a comparison between area and critical path delay has been carried out as shown in Figure 2 by which one can easily visualize optimal performance of the ciphers for instances; `ASCON_GMU` is smaller and faster among all other AEAD ciphers while `COLM` is consuming largest hardware area but its faster than `TIAOXIN_GMU` which is slowest one among all mentioned ciphers.

Figure 4 shows the trend of throughput per KGE of all the AEAD ciphers. Apart from that, power simulation is also computed from design compiler for all the ciphers as shown in Figure 5. Since the critical path delays of all designs in comparison are different for all the ciphers, to ensure that all designs will produce the correct outputs at the same data rate, the clock rate used for the power simulation is chosen based on the slowest design for each. One can notice from Figure 5, `ACORN_8bit` cipher is found to be most power-efficient while `OCB_GMU` design is the most power-hungry among all the designs.

In addition, since faster design generally consumes more dynamic power, to compare the energy consumption of each design running at full speed, power-delay product is computed for each design with the power consumption measured at its maximum allowable clock frequency. Since, each cipher is designed with different size of input data, a comparison has been carried out in terms of throughput per area and throughput per energy as shown in Figure 6. From Figure 6, one can easily find out cipher which has better throughput per area as well as throughput per energy.

4 Cipher Optimization: Architectural

High-level synthesis (HLS) is an increasingly popular approach in electronic design automation (EDA) that raises the abstraction level for designing digital circuits. With the increasing complexity of embedded systems,

Table 3: Synthesis results of AEAD ciphers implementation using TSMC 65nm technology at minimum achievable delay.

Algorithm	Message Bits	Key Size	Tag Size	Throughput (Gbps)	Area (μm^2)	KGE	Throughput per Area	Frequency (Mhz)
AEGIS_GMU	128	32	128	8.65	166109.76	115.35	0.07	540.54
AEZ_GMU	128	32	128	2.98	118654.92	82.40	0.04	581.40
ASCON_GMU	64	32	128	12.88	23504.76	16.32	0.79	1408.45
CLOC_GMU	128	128	128	6.84	140770.8	97.76	0.07	588.24
JAMBU_AES	64	32	64	3.17	38878.92	27.00	0.12	495.05
NORX	768	32	256	57.40	123180.48	85.54	0.67	448.43
OCB_GMU	128	128	128	4.92	144218.52	100.15	0.05	460.83
SILC_GMU	128	32	128	6.40	67765.32	47.06	0.14	500.00
TIAOXIN_GMU	256	32	128	115.32	140689.8	97.70	1.18	450.45
COLM	128	128	128	5.88	177094.8	122.98	0.05	505.05
Deoxys-I	128	32	128	2.43	76846.32	53.37	0.05	549.45
Deoxys-II	128	32	128	4.54	98896.68	68.68	0.07	531.91
Ketje_jr	32	96	64	14.55	17247.96	11.98	1.21	909.09
Ketje_sr	32	128	128	29.09	27694.08	19.23	1.51	909.09
Keyak_lake	1344	128	128	84.85	140949.72	97.88	0.87	757.58
Keyak_river	544	128	128	34.87	70715.52	49.11	0.71	769.23
ACORN_8bit	8	128	128	9.09	9469.8	6.58	1.38	1136.36
ACORN_32bit	32	128	128	34.04	16940.88	11.76	2.89	1063.83
MORUS	256	128	128	281.32	65523.96	45.50	6.18	1098.90

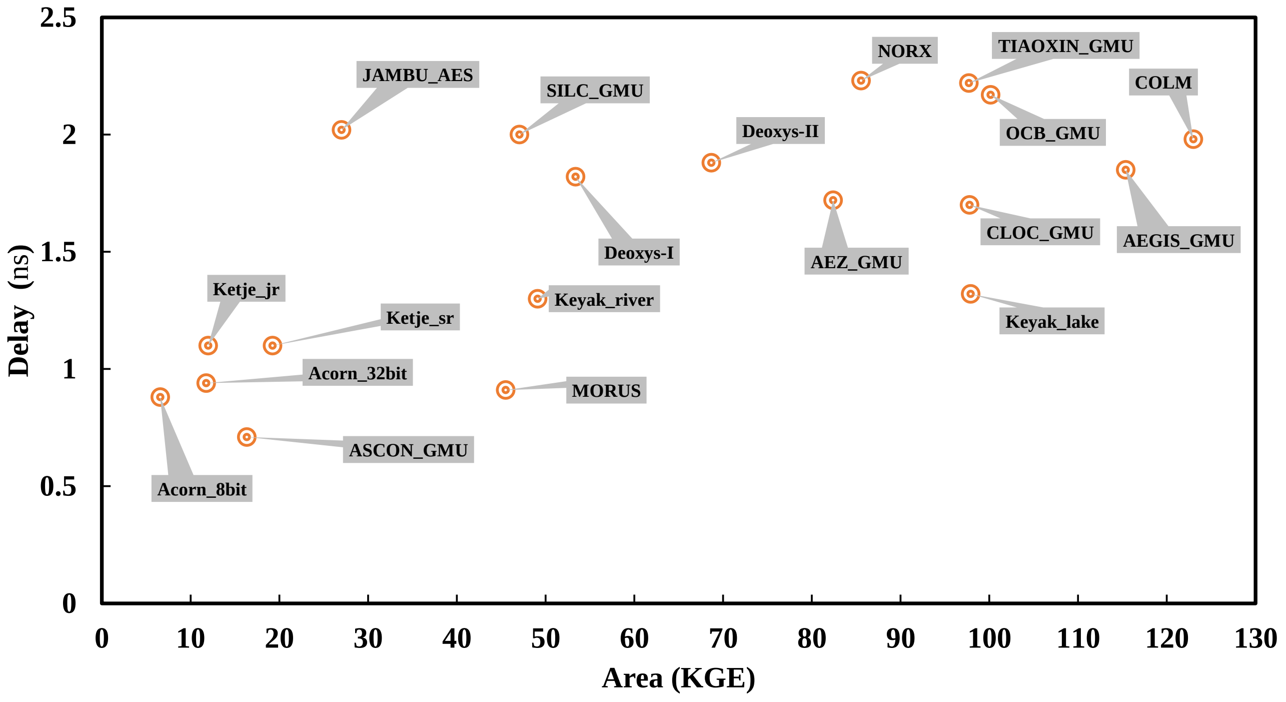

Fig. 2: A comparative analysis of CAESAR third round candidates in terms of synthesis area (KGE) and delay(ns) in ASIC.

these tools are particularly relevant in embedded systems design. HLS proved to be very easy to use for creating a functional RTL design. While in many cases, HLS does not perform well in resource utilization and in many cases it provided a design with a slower maximum clock frequency relative to manual RTL coding[31]. In

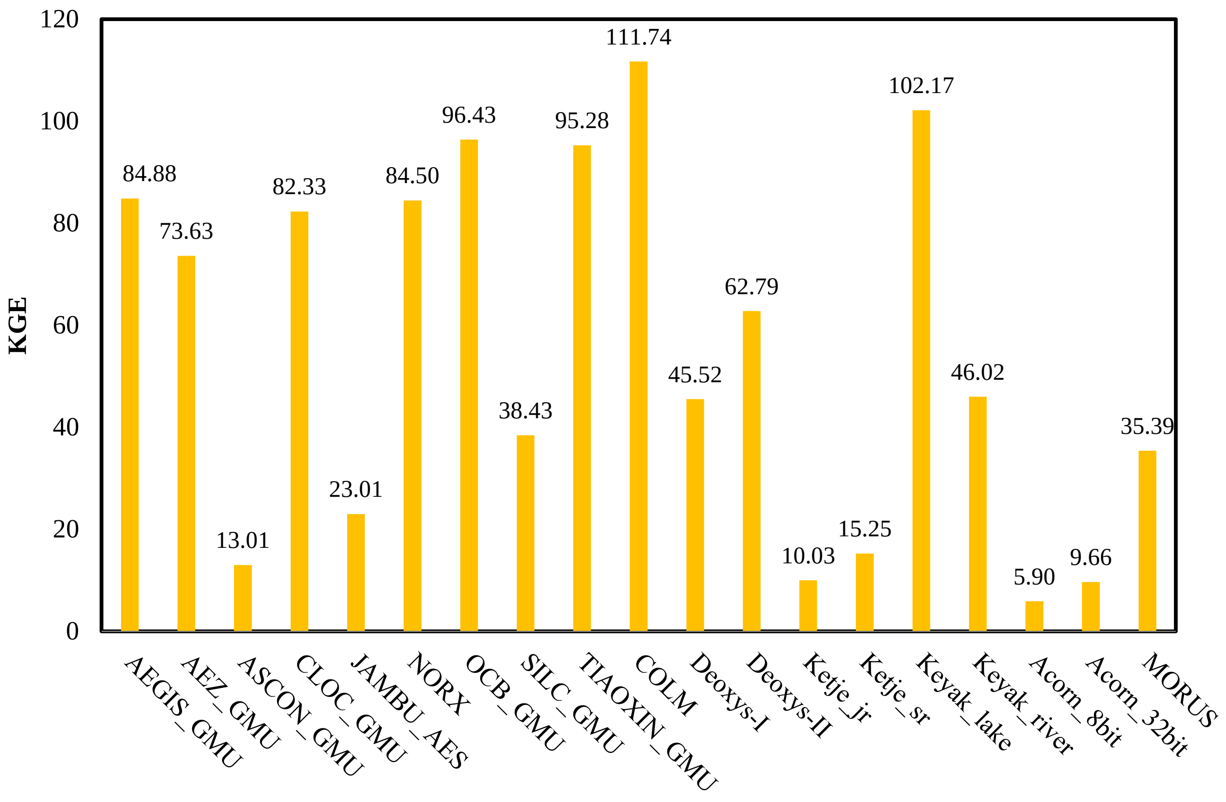

Fig. 3: Synthesis area (KGE) analysis of CAESAR third round candidates at particular frequency (based on the slowest among the ciphers) based on ASIC implementation results.

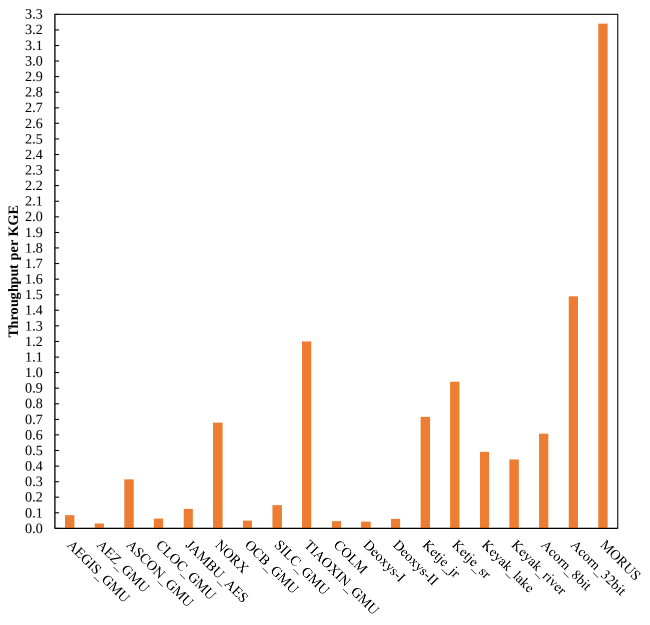

Fig. 4: Analysis of AEAD ciphers in terms of throughput per KGE based on ASIC results at particular frequency (based on the slowest among the ciphers).

the paper [6], the various optimization points for area, speed as well as for throughput are presented for `AEGIS` which shows that there is large room for improvements over automatically generated implementations to improve the performance of AE ciphers selected in the third round of CAESAR. Therefore, in this section, manual optimization is carried out on `NORX`, `CLOC` and `Deoxys-I` based on the provided hardware architecture either from GMU or from designers itself.

4.1 `CLOC`

The automatic generated VHDL code for `CLOC` cipher by high level synthesis tool is not fully optimized and there remains scope of improvement for performance of `CLOC` by removing the redundancy from VHDL code generated from HLS. The undesired `AES` block is removed from `CLOC` design with the help of pipeline technique. Both `CLOC_GMU` and Optimized `CLOC` are implemented in ASIC for minimum achievable delay along with area-constraint. It can be observed from Table 5 that Optimized `CLOC` is 38.30% more energy efficient

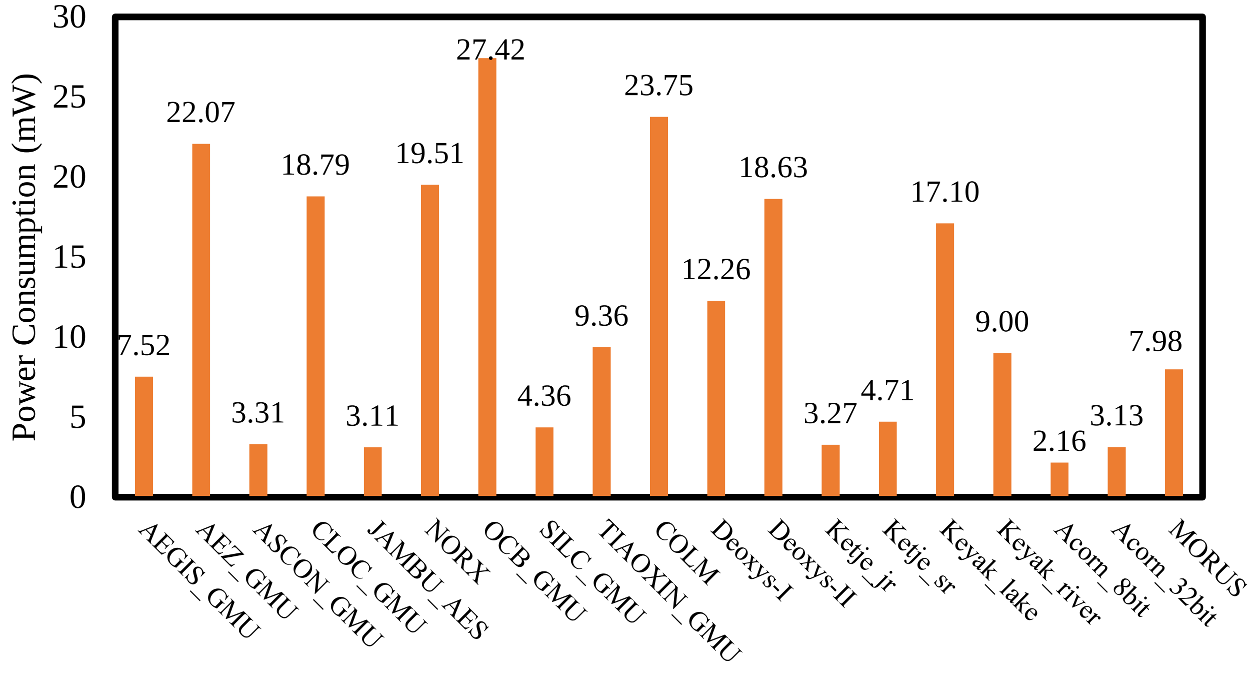

Fig. 5: Power consumption (mW) of AEAD ciphers in ASIC.

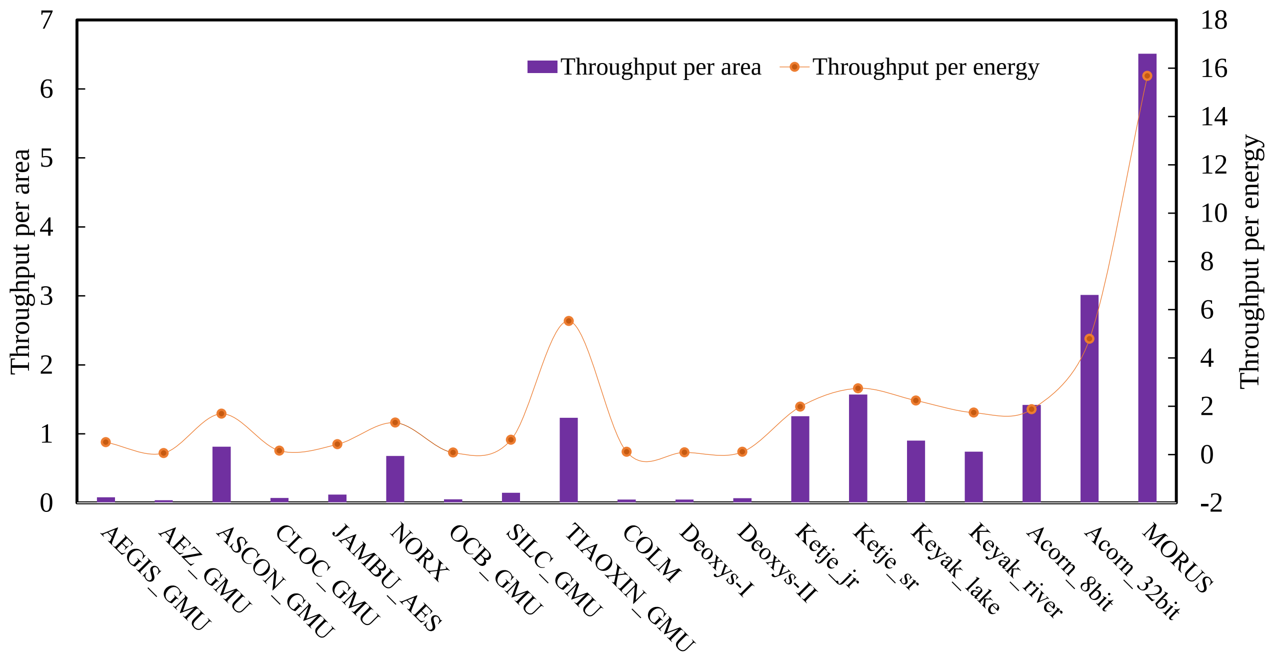

Fig. 6: Analysis of AEAD ciphers in terms of throughput per KGE with respect to throughput per energy based on ASIC synthesis results.

and 20.60% gain in throughput per area when compared with `CLOC_GMU` design.

4.2 `Deoxys`

`Deoxys` is an interesting case study as it belongs to a subset of the CAESAR competition candidates that

Table 4: Synthesis area analysis using TSMC 65nm technology at the minimum frequency (448.43 Mhz).

Algorithm	Message Bits	Key Size	Tag Size	Throughput (Gbps)	Area (μm^2)	KGE	Throughput per Area
AEGIS_GMU	128	32	128	7.17	122228.64	84.88	0.08
AEZ_GMU	128	32	128	2.30	106034.04	73.63	0.03
ASCON_GMU	64	32	128	4.10	18739.08	13.01	0.32
CLOC_GMU	128	128	128	5.22	118548	82.33	0.06
JAMBU_AES	64	32	64	2.87	33141.24	23.01	0.12
NORX	768	32	256	57.40	121675.68	84.50	0.68
OCB_GMU	128	128	128	4.78	138865.33	96.43	0.05
SILC_GMU	128	32	128	5.74	55336.68	38.43	0.15
TIAOXIN_GMU	256	32	128	114.80	137198.16	95.28	1.20
COLM	128	128	128	5.22	160903.44	111.74	0.05
Deoxys-I	128	32	128	1.98	65542.32	45.52	0.04
Deoxys-II	128	32	128	3.83	90419.04	62.79	0.06
Ketje_jr	32	96	64	7.17	14439.6	10.03	0.72
Ketje_sr	32	128	128	14.35	21954.24	15.25	0.94
Keyak_lake	1344	128	128	50.22	147122.28	102.17	0.49
Keyak_river	544	128	128	20.33	66272.4	46.02	0.44
ACORN_8bit	8	128	128	3.59	8494.92	5.90	0.61
ACORN_32bit	32	128	128	14.35	13914.72	9.66	1.49
MORUS	256	128	128	114.80	50965.55	35.39	3.24

Table 5: Performance analysis of CLOC_GMU and Optimized CLOC implemented by using TSMC 65nm technology.

AEAD Ciphers	Delay (ns)	Frequency (Mhz)	Area (μm^2)	KGE	Energy (pJ)	Throughput (Mbits/s)	Throughput per Area
Optimized CLOC	1.34	746.26	96605.28	67.09	26.25	2850.75	29.50
CLOC_GMU	1.70	588.23	136618.56	94.87	42.55	3341.18	24.46
Improvement	21.17%	26.86%	29.29%	29.29%	38.30%	-14.67%	20.60%

are amenable to parallelisation, i.e. different message blocks can be processed in parallel. This subset also include candidates such as: AEGIS, COLM, MORUS and OCB. Four of these five candidates are AES-based (except MORUS), so it is believed that any significant improvement to the implementation of any of these algorithms will also benefit the other three algorithms. We have implemented the Deoxys-I-128 4-stream architecture from [22] with full compliance to the CAESAR Hardware API. While that architecture is originally targeted towards FPGA implementations, it was found that it also enhances the ASIC results, leading to 11.5% smaller area, 54.3% higher speed and 38.3% higher efficiency. The significance of these results is that it shows that parallelisation should be considered while performing benchmarking of different candidates. Besides, it shows that the FPGA and ASIC are not matching, as the API compliance reduces the performance of this architecture on FPGA as mentioned in [22], but the ASIC design tools where able to optimize the API control unit to keep up with the high speed data path.

4.3 NORX

In order to show the optimization potential for the automatically generated code of the ciphers, we have performed hardware optimization for the NORX cipher as well with manual changes.

On of the issues that we have observed with the automatically generated NORX hardware implementation is the resources duplication, for example, when in theNORX algorithm, the same operation is performed (called G function) on both columns and diagonals sequentially (i.e. $S \leftarrow diag(col(S))$), the function G was duplicated many times at the automatically generated code. The

Table 6: Synthesis results of the `Deoxys-I-128` implementation using TSMC 65nm technology

Impl.	Area (KGE)	Max. Freq. (Mhz)	Throughput (Mbps)	Efficiency (Mbps/KGE)
Optimized `Deoxys`	59.53	847	7,227	121.40
`Deoxys_GMU`	53.37	549	4,684	87.76
Improvement	-11.5%	54.3%	54.3%	38.3%

optimization that we have entered to fix this issue is pipelining. The optimization removes half of the G function instances by converting the implementation to be sequential, a buffer was added to collect the intermediate results and route them to the existing resources. In addition, a small finite state machine (FSM) was added in order to sequence the flow. In order to account for the additional delay due to the insertion of the pipe stages, few counters that controls the main pipeline of the `NORX` were modified.

Table 7 shows that with the optimized version of `NORX`, the Throughput is 44.8% higher than the original `NORX` while the area was reduced by 18%. Furthermore, it can be seen that the throughput per area metric was improved by 76.9% at the optimized `NORX` compared to the standard `NORX` architecture.

The synthesis area of `NORX` is further reduced by 36.7% when performing the simulation at minimum frequency (maximum time between both designs). The efficiency (throughput per area) of optimized `NORX` is improved by 37.3%.

4.4 Comparison between ASIC based results and FPGA based results

In addition, to have performance comparison in ASIC, optimized `NORX` and `CLOC` have also been implemented on FPGA platform (Virtex 6). In Table 9, the pre-layout results based on FPGA of both ciphers are tabulated along with its contenders. Optimized `NORX` is 39.13% faster and improved 30% throughput per area, 4% gain in throughput but requires 8% more area when compared with its contender while, based on ASIC results, optimized `NORX` achieves 44.8% gain in throughput, 18% reduction in area and throughput per area metric improved by 76.9%. Similarly, based on FPGA result, optimized `CLOC` is faster by 3.4%, reduction in area by 5.4% but throughput per area is reduced by 26.42%. ASIC based results show that optimized `CLOC` outperformed in terms of area, speed and throughput per area. These results prove that synthesis results based on FPGA are different from the results obtained from ASIC technology library. In addition, the analysis of throughput per area based on ASIC results as well as on FPGA results has been carried out for third round of CAESAR candidates in Figure 7.

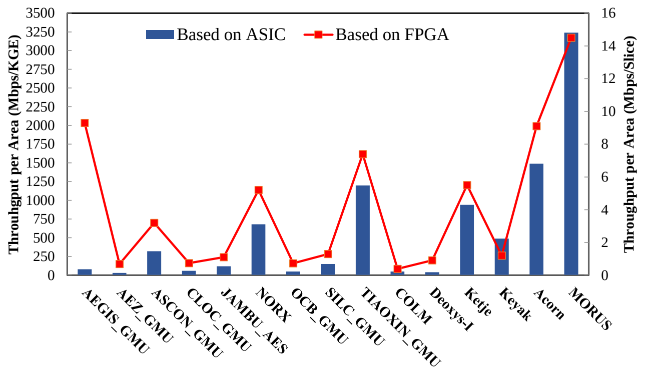

Fig. 7: Throughput per area comparison based on ASIC results and FPGA results.

5 Cipher Optimizations: Technology-Aware

Another factor of hardware benchmarking is technology-mapping and/or technology aware optimization. For example, optimizing architectures and circuits for ASIC implementations uses different techniques as opposed to other technologies such as FPGA. The main reason is that FPGA does not use logic gates as the building unit. It uses fixed size (e.g. 6-to-1) LUT-FF pairs. This issue has been extensively studied in [22]. The results showed that the efficiency of AES FPGA implementations can be almost doubled by using FPGA-specific optimization techniques. Again, this optimization had to be manually implemented as the HLS synthesis tools and the FPGA RTL optimization tools are not short enough to invoke them. However, the current trend in cryptographic hardware benchmarking, specially in the CAESAR competition is to use a ”one size fits all” approach, where most of the time only one circuit is developed per algorithm and used for all technologies.

Table 7: Synthesis results of Optimized-NORX and NORX [13] implemented using TSMC 65nm technology at minimum delay.

AEAD Ciphers	Delay (ns)	Frequency (Mhz)	Throughput (Gbps)	Area (μm^2)	KGE	Throughput/Area
Optimized NORX	1.32	757.57	83.11	100988.64	70.13	1.185
NORX [13]	2.23	448.43	57.40	123180.48	85.54	0.67
Improvement	40.8%	68.9%	44.8%	18.0%	18.0%	76.9%

Table 8: Synthesis area of Optimized NORX and NORX at 448.43 Mhz based on ASIC results.

AEAD Ciphers	Throughput (Gbps)	Area (μm^2)	KGE	Throughput/Area
Optimized NORX	49.20	77057.28	53.51	0.92
NORX [13]	57.40	121675.68	84.50	0.67
Improvement		36.7%	36.7%	37.3%

Table 9: Synthesis results based on FPGA of NORX and CLOC

	Design	Max. Freq.	Area(slice)	Throughput(Mbps)	Throughput /Area (Mbps/slice)
NORX	Optimized NORX	320	2398	40,960	17.09
	NORX [13]	230	2226	29,440	13.22
	Improvement	39.13%	-8%	39.13%	29.3%
CLOC	Optimized CLOC	182	595	695.24	1.17
	CLOC_GMU [13]	176	629	999.68	1.59
	Improvement	3.4%	5.41%	-30.45%	-26.42%

One of the examples where FPGA Technology-Aware design can be used to achieved huge efficiency gains is the pipeline retiming. Pipelining has been used by hardware designers/architects as a tool to increase throughput/run-time performance for a long time. However, a fully pipelined block cipher implementations can be costly, due to the large area requirements. A more realistic approach is to use multi-stream implementations. These implementations start from a sequential implementation that processes one block in C cycles, and divides it into N pipeline stages. This leads to computing x blocks in $N \cdot C$ cycles, where $x \in \{1, 2, ..., N\}$. x depends on the number of independent block streams the user can leverage. However, this is a double-edged weapon, due to the following reasons:

1. The time required to process one block in a sequential implementation is $\sim C{\cdot}T$, where T is the critical path delay of the implementation. If the N pipeline stages divide the critical path evenly into segments of $\frac{T}{N}$ delay, the time required to process N blocks becomes $T{+}t$, where t is a small overhead, leading to $\sim N$x speed-up. Unfortunately, the critical path is usually not evenly divided, leading to a sub-optimal speed-up ($< N$).
2. Modern FPGA families consist of a basic building block called LUT6, which is a 6-input single-output look-up table. Additionally, each unit of this building block has an associated Flip-Flop, which the designer/synthesis tool can choose to either use it or not. In Figure 8, we show the optimal utilization of a LUT6 unit in a pipelined architecture, where it is used to implement a 6-input circuit followed by storing the output. On the other hand, in Figure 9, a poor selection of the location of pipeline stage is in-place, leading to the utilization of 3 look-up tables, instead of 1 in the case of Figure 8. In other words, the poor choice of where to place the pipeline registers leads to a significant increase in area.

The effect of this technique, among others, has been used to develop the most efficient FPGA implementations for both AES and LED block ciphers, as reported in [22], achieving 1.9× and 2.57× efficiency gains over the previous results, respectively.

While it is believed that such techniques are currently known to a wide variety of hardware design-

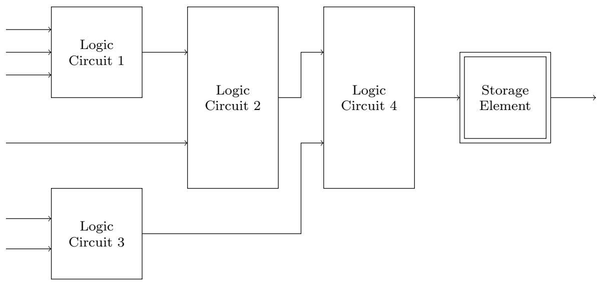

Fig. 8: Optimal pipeline selection

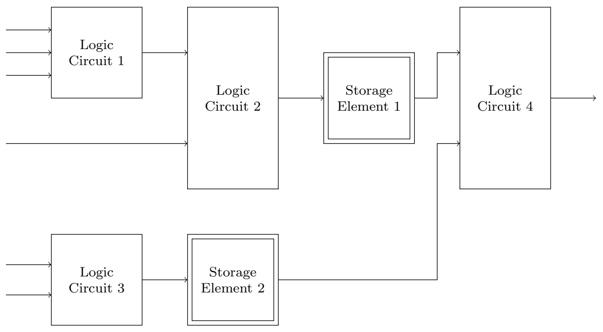

Fig. 9: Sub-optimal pipeline selection

ers, the reason of adapting this approach is the lack of enough work-force to design circuits for the increasing of ciphers being proposed. This is one of the reasons that motivated the work in this paper, and we encourage more hardware designers and research groups to join this process. It is also expected that in the next round of the competition the number of candidates will be smaller, which will allow more extensive studies on the candidates. For example, besides the technology aware benchmarking, lightweight implementations are not yet available for most of the candidates, as well.

6 Influence of CAESAR API on the Performance

6.1 What is API?

An Application Programming Interface (API) is a set of subroutine definitions, protocols, interfaces and tools for building application. In general terms, it is a set of clearly defined methods of communication between various components, while the components can be hardware, software or firmware. A good API makes it easier to develop a system or sub-system by providing all the building blocks, which are then put together by the developer.

The CAESAR Hardware API [15] is intended to meet the requirements of all algorithms submitted to the CAESAR competition, as well as many earlier developed authenticated ciphers. The main purpose of the API is to guaranteeing compatibility among implementations of the same algorithm developed by different designers, and to guaranteeing a fair benchmarking of authenticated ciphers in hardware.

6.2 Structure of the CAESAR Hardware API

The top-level block diagram of a high-speed, non-pipelined implementation of any authenticated cipher compliant with the CAESAR hardware API is shown in Figure 10.

The top-level unit, called AEAD, is divided into four lower-level units, called, *PreProcessor*, *PostProcessor*, *CMD FIFO*, and *Cipher-Core*. The codes for the first three units are provided as part of the Development Package [10] of the CAESAR benchmarking project. Due to the availability of this package as well as the well-defined hardware API of the CipherCore the developers can focus on implementing the *CipherCore* itself while not concerned with the internal details of the *PreProcessor*, *PostProcessor*, and *CMD FIFO*.

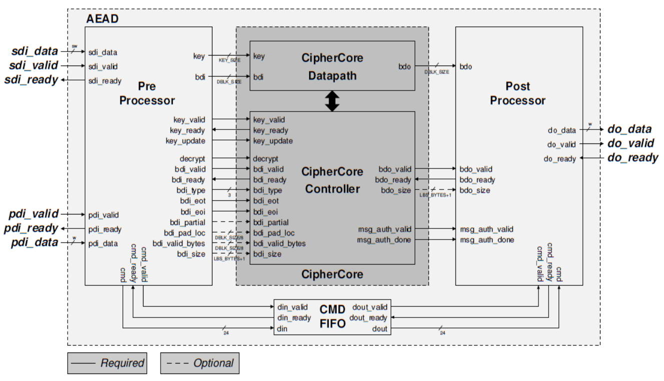

Fig. 10: Top-level block diagram of AEAD architecture for CAESAR [13]

6.3 Discussion on the CAESAR Hardware API

While the benchmarking efforts conducted in this paper or in [16] have used only implementation that are compliant with the CAESAR Hardware API, there are several clues that this may not be the optimal approach. For example, in [22] the authors have shown that with a custom interface and control unit, parallelism can increase the efficiency of `Deoxys-I-128` on FPGA by 75%. Table 10 show the comparison between the implementations without the API support, while Table 11 shows the effect of API support on the 4-steam architecture in [22]. It can be seen that the API support cuts the performance and efficiency almost to by 50%, due to the strict requirements on the control unit design. However, with the API support, the design becomes crippled due to the strict requirements on how the input is handled and how the software/hardware interface is implemented. Additionally, the API favors a certain type of implementations (basic iterative implementations). The control unit becomes huge when more control requirements are added to the design due to architectural optimization, such as parallelized/pipelined architectures or 8/32-bit data paths.

One example where the API adds a lot of redundant cost to the implementation is processing the length of the message. Many AEAD ciphers do not require any knowledge about the message length in advance. They only require knowledge of the last block. This is 1-bit worth of information that can be very efficiently communicated as part of the input. However, the only way the API allows this information to be communicated is by sending the message length as part of the message header, which enforces the need for a huge counter inside the control unit to detect when the last block is reached.

In addition, [19,2] provided very small ASIC implementations for AES, including both the encryption and decryption circuits. Both circuits have area of around 2000 GEs. While such implementations provide a proof-of-concept that AES-based ciphers can have a very small footprint, these circuits cannot be directly incorporated in the CAESAR benchmarking process, as they handle the bytes inside every data block in and "exotic" order, which is not allowed by the CAESAR Hardware API. While we believe that indeed such ordering may cause compatibility issues if the communicating devices do not follow the same ordering, there are still a wide variety of applications where the devices are developed by the same manufacturer and configured by the same user. One example of such applications is Wireless Sensor Networks (WSNs) and RFID Tags/Readers. These applications can benefit a lot from a small area circuit, yet such scenarios are not captured by the current benchmarking process.

Moreover, the current benchmarking process [13] is targeted only to a specific technology, namely FPGA. In [16], the authors mentioned that one implementation was not considered for benchmarking for using clock gating, which is an ASIC design technique. Similarly, many design techniques can be excluded. Hence, we believe that both ASIC and FPGA should be considered for the competition.

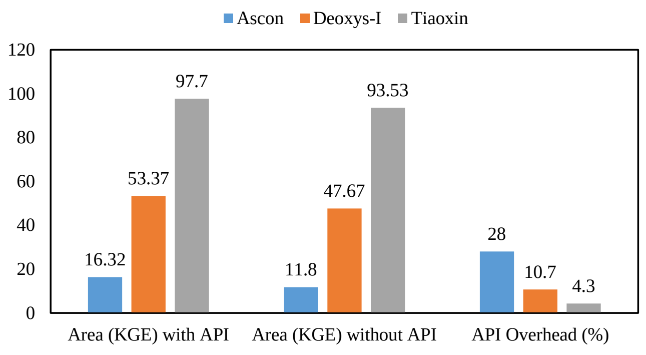

Fig. 11: Area vs. API overhead for ASCON, Deoxys-I and TIAOXIN

Another implementation have been rejected for sharing the same input port between both the secret and public data, aiming at more compact hardware and less number of pins. The implementation violates an essential requirement of the CAESAR Hardware API. However, the trade-off between pin-limited vs. logic-limited IC design is an important issue in hardware design. There are many applications where the final chip is not bounded by the amount of logic inside, but by the large number of pins used. Such chips under-utilize the area just to accommodate the pins required. Hence, a fixed API may not be justified from the overall IC design viewpoint.

On the other hand, for the same architecture, such as basic iterative implementations, the API cost is almost constant. Our ASIC results show that the area overhead due to the API support is between 4 and 5 KGE. Consequently, the effect of the API on area is more crucial for small designs, such as ASCON, while it is negligible for designs with large area, such as TIAOXIN. To illustrate this phenomenon, figure 11 shows a simple comparison between the area of the implementations of three ciphers, ASCON (small), Deoxys-I (medium) and TIAOXIN (large) and the area overhead due to the API on each of them. The impact of the Hardware API on the area/performance on FPGA has been also discussed in [20]. The authors showed that the GMU Hardware API used in the SHA-3 competition hardware benchmarking, can cause 25% overhead in terms of the area compared to other interfaces they have provided. The CAESAR Hardware API is an updated version of the GMU Hardware API.

Having explained that, the API has a lot of advantages, such as ensuring that the cost of key scheduling is included in the benchmarking process and having a limit for the number of pins used. However, based on the results of the study in this paper, it is advisable to have more flexibility in developing hardware implemen-

Table 10: `Deoxys-I`-128 implementation on Virtex 6 with custom interface

Impl.	Number of Slices	Max. Freq. (MHz)	Throughput (Mbps)	Efficiency (Mbps/Slice)
[22]	861	454	3,874	4.5
[13]	946	285	2,432	2.57

Table 11: `Deoxys-I`-128 implementation on Virtex 6 with API Support

Impl.	Number of Slices	Max. Freq. (MHz)	Throughput (Mbps)	Efficiency (Mbps/Slice)
[22]	805	225	1,920	2.38
[13]	956	285	2,432	2.54

tations, in order to tweak the control unit to suit every algorithm or architecture. Figure 12 shows the trend between area in KGE and delay (ns) of `NORX`, Optimized `NORX`, `CLOC_GMU`, optimized `CLOC`, Optimized `Deoxys-I`, `ASCON_GMU` (with or without API), `TIAOXIN_GMU` (with or without API)and `Deoxys-I` (with or without API) based on ASIC results.

Summary

In this paper, a comprehensive study regarding the hardware evaluation of AEAD ciphers has been carried out with the help of ASIC standard technology library. The benchmarking based on ASIC implementation results has been reported in terms of area (KGE), frequency (Mhz), throughput(Mbps or Gbps), throughput per area etc. This paper also shows that design based on high level synthesis tool does not utilize all optimization technique. Thus, manual optimization is required. In order to support this, the ciphers `NORX`, `Deoxys-I` and `CLOC` have been manually optimized. The experimental result shows that optimized `NORX`, `Deoxys-I` and `CLOC` outperformed in terms of area, delay, throughput per area and energy efficiency when compared with its respective contenders.

The comparison has been made between ASIC based results and FPGA results. At the end, influence of CAESAR APIs on the overall performance of cipher has also been analyzed. For instance; Deoxys-I-128 cipher, supporting CAESAR API cuts the performance and efficiency almost to by 50% compared to custom interface, due to the strict requirements on the control unit design.

References

1. Abed, F., Forler, C., Lucks, S.: General classification of the authenticated encryption schemes for the CAESAR competition. Computer Science Review (2016)
2. Banik, S., Bogdanov, A., Regazzoni, F.: Atomic-AES: A Compact Implementation of the AES Encryption/Decryption Core. In: Progress in Cryptology–INDOCRYPT 2016: 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings 17, pp. 173–190. Springer (2016)
3. Bellare, M., Rogaway, P., Wagner, D.: A conventional authenticated-encryption mode. manuscript, April (2003)
4. Bernstein, D.J.: Failures of Secret Key Cryptography. Invited talk at FSE (2013)
5. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: In A. Miri and S. Vaudenay, editors, pp. 320–337. Springer (2011)
6. Bhattacharjee, D., Chattopadhyay, A.: Efficient Hardware Accelerator for AEGIS-128 Authenticated Encryption, pp. 385–402. Springer International Publishing, Cham (2015)
7. Bogdanov, A., Knudsen, L., Leander, G., Paar, C., Poschmann, A., , Robshaw, M., 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 (2007)
8. Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In: International Symposium on Experimental Algorithms, pp. 178–189. Springer (2010)
9. Boyar, J., Peralta, R.: A Small Depth-16 Circuit for the AES S-Box. In: IFIP International Information Security Conference, pp. 287–298. Springer (2012)
10. CAESAR Competition: CAESAR submissions. `https://competitions.cr.yp.to/caesar-submissions.html` (2016)
11. Canniere, C.D., Preneel, B.: Trivium: In New Stream Cipher Designs The eSTREAM Finalists. In: Springer verlag. Springer (2008)
12. Canright, D.: A Very Compact S-box for AES. In: International Workshop on Cryptographic Hardware and Embedded Systems, pp. 441–455. Springer (2005)
13. George Mason University: ATHENa: Automated Tools for Hardware EvaluatioN. `https://cryptography.gmu.edu/athena/` (2017)

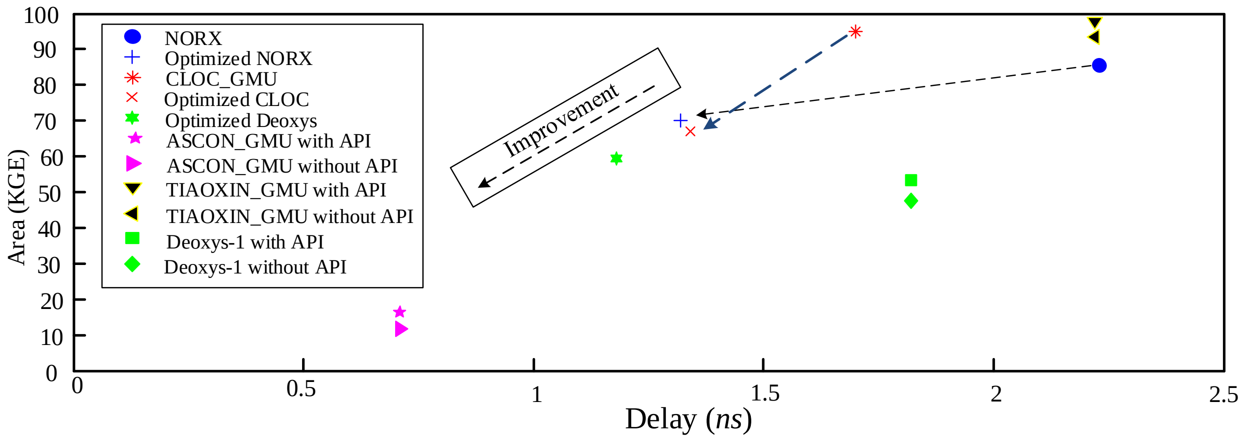

Fig. 12: The comparison between area's and delay's of `NORX`, Optimized `NORX`, `CLOC_GMU`, optimized `CLOC`, `Deoxys-I` (with or without API), Optimized `Deoxys-I`, `ASCON_GMU` (with or without API) and `TIAOXIN_GMU` (with or without API) respectively based on ASIC results.

14. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: International Workshop on Cryptographic Hardware and Embedded Systems, pp. 326–341. Springer (2011)
15. Homsirikamol, E., Diehl, W., Ferozpuri, A., Farahmand, F., Yalla, P., Kaps, J.P., Gaj, K.: CAESAR Hardware API. Cryptology ePrint Archive, Report 2016/626 (2016)
16. Homsirikamol, E., Farahmand, F., Diehl, W., Gaj, K.: Benchmarking of Round 3 CAESAR Candidates in Hardware: Methodology, Designs & Results. `https://cryptography.gmu.edu/athena/presentations/CAESAR_R3_HW_Benchmarking.pdf` (2017)
17. Homsirikamol, E., Gaj, K.: An Alternative Approach to Hardware Benchmarking of CAESAR Candidates Based on the Use of High-Level Synthesis Tools. `https://cryptography.gmu.edu/athena/presentations/GMU_DIAC_2016_HLS.pdf` (2016)
18. Information technology – Security techniques – Authenticated encryption. Standard, International Organization for Standardization (2009)
19. Jean, J., Moradi, A., Peyrin, T., Sasdrich, P.: Bit-Sliding: A Generic Technique for Bit-Serial Implementations of SPN-based Primitives. In: International Conference on Cryptographic Hardware and Embedded Systems, pp. 687–707. Springer (2017)
20. Jungk, B., Stttinger, M.: Hobbit - Smaller but faster than a dwarf: Revisiting lightweight SHA-3 FPGA implementations. In: 2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp. 1–7 (2016). DOI 10.1109/ReConFig.2016.7857176
21. Jutla, C.S.: Encryption Modes with Almost Free Message Integrity. Cryptology ePrint Archive, Report 2000/039 (2000). `http://eprint.iacr.org/2000/039`
22. Khairallah, M., Chattopadhyay, A., Peyrin, T.: Looting the LUTs : FPGA Optimization of AES and AES-like Ciphers for Authenticated Encryption. The 18th International Conference on Cryptology in India - IndoCrypt (2017)
23. Kohno, T., Viega, J., Whiting, D.: The CWC authenticated encryption (associated data) mode. ePrint Archives (2003)
24. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: International Workshop on Fast Software Encryption, pp. 306–327. Springer (2011)
25. Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In: EUROCRYPT, pp. 275–292. Springer (2014)
26. Peyrin, T., Seurin, Y.: Counter-in-Tweak: authenticated encryption modes for tweakable block ciphers. In: Annual Cryptology Conference (2016)
27. Resende, J.C., Chaves, R.: AES Datapaths on FPGAs: A State of the Art Analysis. In: Hardware Security and Trust, pp. 1–25. Springer (2017)
28. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 16–31. Springer (2004)
29. Rogaway, P., Bellare, M., Black, J.: OCB: A Block Cipher Mode of Operation for Efficient Authenticated Encryption. ACM Transactions on Information and System Security (TISSEC) **6**(3), 365–403 (2003)
30. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: A Lightweight Block Cipher for Multiple Platforms. In: Selected Areas in Cryptography, pp. 339–354. Springer (2012)
31. Zwagerman, M.D.: High level synthesis, a use case comparison with hardware description language (2015)