Evaluation of the CAESAR Hardware API for Lightweight Implementations

Panasayya Yalla, Jens-Peter Kaps
Department of Electrical and Computer Engineering, George Mason University, Fairfax, Virginia 22030, USA

Cryptographic Engineering Research Group

Abstract

The Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) requires that all hardware implementations of candidate algorithms adhere to the CAESAR Hardware API [1]. The CAESAR Hardware API is supported by a development package which includes VHDL code for universal pre- and post-processors for high-speed and recently also for lightweight implementations. These processors are designed to make a cipher core compliant with the API. In this work we verify that the lightweight package has a smaller area footprint than the high-speed package. We also show that the overhead of using the generic lightweight pre- and post-processors over integrating their functionality into the cipher core is negligible. As part of these case studies, we have developed the first lightweight implementations of KETJE-SR, ASCON-128, and ASCON-128a.

Introduction and Motivation

- CAESAR evaluates candidates for a final portfolio of new Authenticated Encryption with Associated Data (AEAD) algorithms.
- All candidates must adhere to the CAESAR hardware (HW) Application Programming Interface (API).
- The HW API is one component which enables a fair comparison among algorithms.
  - Independent FIFO inputs for public data (PDI) and secret data (SDI) and FIFO output (DO).
  - In-band signaling for commands and data types using a simple protocol.
- CAESAR HW API is supported by an implementer's guide and development package [2].
  - Includes VHDL code for high-speed (HS) and lightweight (LW) implementations.
  - Pre- and PostProcessor separate protocol from cryptographic algorithm.
  - Bypass FIFO stores and passes header information to PostProcessor.
- It is generally assumed that having generic pre-and post-processors increases the area consumption over merging their functionality with the cipher cores.

Differences between HS vs. LW Packages

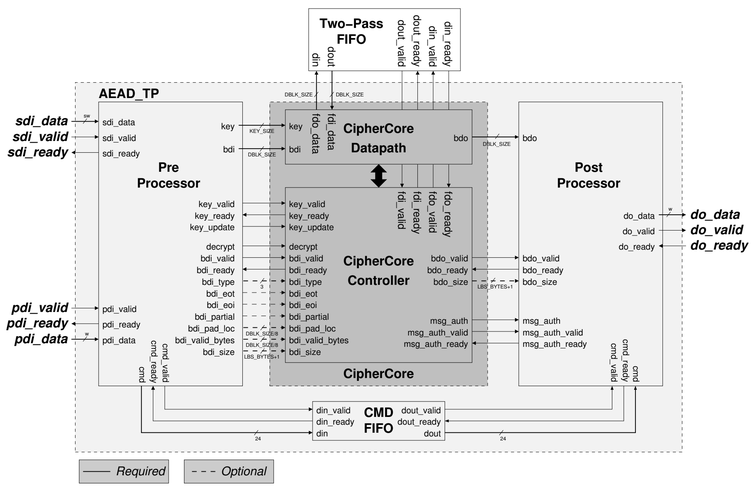
High-Speed

- Supports bus width $32 \leq w \leq 256$ in multiples of 8.
- PreProcessor expands PDI and SDI data to full block size for CipherCore.
- PreProcessor stores one block of PDI and SDI data.
- PreProcessor contains universal padding unit.
- Tag comparison has to be performed in CipherCore.

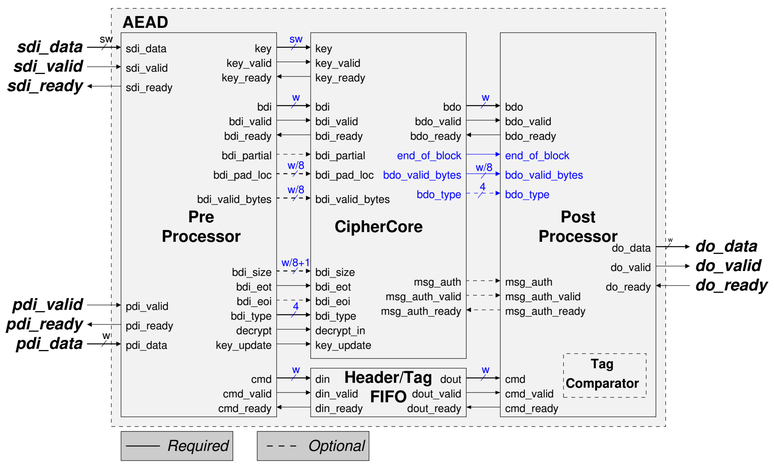
Lightweight

- Supports bus width w of 8, 16, and 32.
- PreProcessor, CipherCore, Bypass FIFO, and PostProcessor have equal bus width.
- PreProcessor has no data storage.
- Assumes padding is performed in CipherCore.
- PostProcessor supports tag comparison.

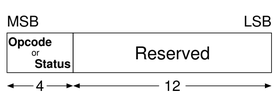
CAESAR High-Speed Block Diagram



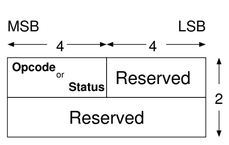
CAESAR Lightweight Block Diagram



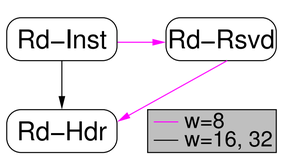
Protocol: Instruction



16-bit Instruction with *w=16*

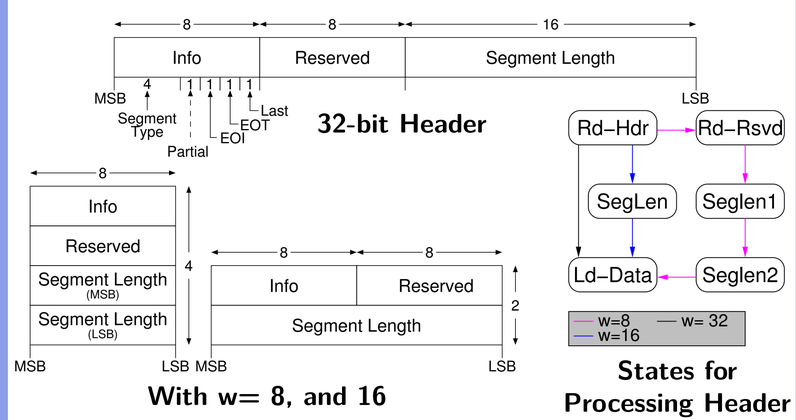


16-bit Instruction with *w=8*



States for Processing Instruction

Protocol: Segment Header



Case Study

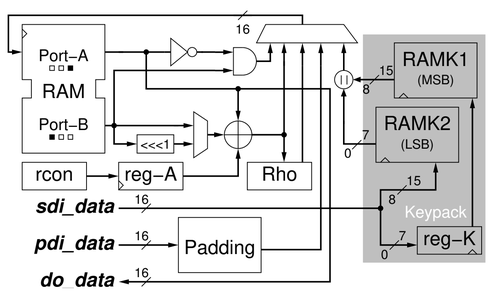
1. Determine overhead of CAESAR LW package.
   - KETJE-SR implementation with integrated support of CAESAR API.
   - KETJE-SR implementation using new CAESAR lightweight development package.
2. Determine overhead of CAESAR LW package vs HS package.
   - Implementation of ASCON using CAESAR LW package.
   - Using existing ASCON HS implementation.

KETJE-SR

- Ketje [3] is based on round reduced Keccak-f called MonkeyWrap.
- Has four variants Ketje-Jr, Ketje-Sr, Ketje-Minor, and Ketje-Major which use Keccak-p^*[200], Keccak-p^*[400], Keccak-p^*[800], and Keccak-p^*[1600] respectively.
- Each round of Keccak-p^* consists of five steps θ, ρ, π, χ, and ι.
- In θ step, each bit in the state is Xored with two other bits from two different columns.
- The state bits are rotated for each lane using one of the 25 different offsets in ρ step
- Lanes are rearranged in π, integer multiplication in χ.
- The last step is ι, where a round constant is added.

KETJE-SR Datapath

- We implemented a Ketje-Sr using a 16-bit datapath and interface.
- Datapath is the same for integrated CAESAR API support and using CAESAR LW package.
- State is stored in a dual-port memory (RAM) with one read/write and one read-only ports.
- To reduce the complexity of padding for key, the key size is fixed to 128-bits.
- Two memory units (RAMK1, and RAMK2) with pre-stored values and a register (reg-K) for key storage and *KeyPack* operations.
- Padding for message and AD using multiplexers.
- Needs 160 clock cycles to process a 32-bit block.
- $TP = \frac{32}{160} \cdot F$

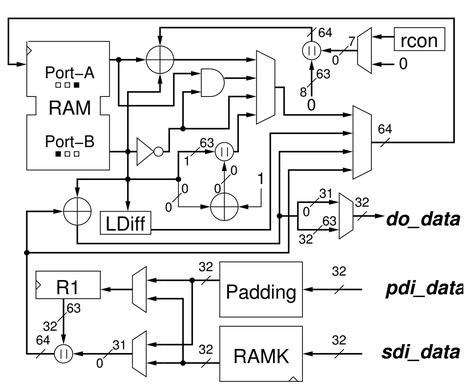


ASCON

- ASCON[4] is a permutation based authenticated cipher.
- ASCON-128, and ASCON-128a - two variants with block sizes of 64 and 128 respectively.
- In each round, three sub transformations called constant-addition, substitution, and linear diffusion are applied.
- Constant-addition is the first operation in the round, where a constant is added to one of the five words. Twelve round constants are used.
- Substitution layer uses 5x5 S-boxes.
- Linear diffusion layer for diffusion across each of the five 64-bit words using circular shifts and an *XOR*.

ASCON Datapath

- A 64-bit datapath is used in this design and a 32-bit interface.
- The state is stored in a dual-port RAM.
- Key is stored in a RAM.
- The substitution layer is implemented in a bit-slice fashion.
- Due to the contention on the RAM ports, 18 operations take 33 clock cycles.
- The round constants are generated using two 4 bit registers and adders.
- Two 5-to-1 multiplexers are used to perform circular shifts in linear diffusion step (LDiff).
- $TP = \frac{128}{33 \cdot 8} \cdot f$



Case Study 1: Integrated vs. LW Package

Implementation Results on Xilinx Spartan-6 FPGA using ATHENa [5]

Design	Slices	LUTs	Flip-Flops	Freq [MHz]	TP [Mbps]	TP/Area [Mbps/slice]
KETJE-SR[1]	140	436	98	122.4	24.48	0.17
KETJE-SR[2]	155	450	114	120.1	24.03	0.16
Overhead	15	14	16			
ASCON-128[2]	231	684	268	216.0	60.10	0.26
ASCON-128a[2]	231	684	268	216.0	119.16	0.52
Joltik [6][3]	168	534	381	200.0	426.67	2.54
ACORN [6][4]	202	540	383	231.6	1,852.80	9.17

[1] ⇒ Dedicated CAESAR API; [2] ⇒ CAESAR LW Package; [3] ⇒ Not compliant to CAESAR API; [4] ⇒ Tweaked CAESAR HS Package

- Using CAESAR LW Package leads to a small area increase.
  - Three separate counters for *sdi*, *pdi* and *do* buses are used for simplicity and parallel operation.
  - Counter for *sdi* can be dropped if cipher core provides end_of_key signal.
- Comparing our designs which each other and other reported implementations.
  - ASCON-128a has 4 times the TP while consuming only 50% more slices.
  - Joltik implementation is not compliant with CAESAR API but performs significantly better.
  - ACORN is based on a stream cipher which typically perform very well in lightweight implementations.

Case Study 2: Area Overhead HS vs. LW Pkg.

Area Overhead High-Speed (HS) vs. LightWeight (LW) Packages
Implementation Results on Xilinx Spartan-6 FPGA using ATHENa [5]

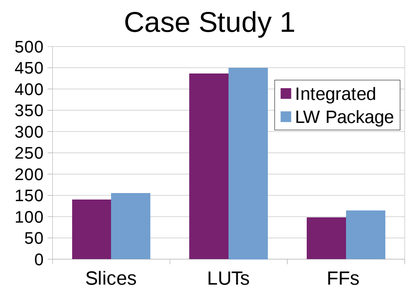
Design	Top-level	Slices	LUTs	Filp-Flops
LW ASCON	AEAD[1]	231	684	268
	CipherCore	196	606	212
Overhead		35	78	56
HS ASCON [6]	AEAD[2]	416	1282	792
	CipherCore	379	1033	529
Overhead		37	249	263

[1] ⇒ CAESAR LW Package; [2] ⇒ CAESAR HS Package

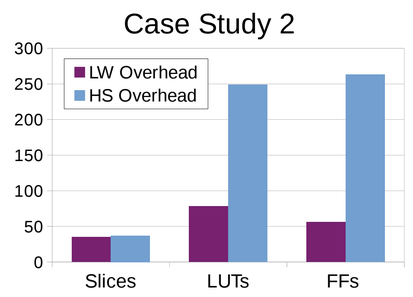
- Adding CAESAR API support to
  - LW core using LW Package leads to a small area increase,
  - HS core using HS Package leads to a larger area increase.

Conclusions

- The graph shows implementation results of KETJE-SR on Spartan-6.
- Using the CAESAR LW Package leads to a small area increase over integrated designs.
- This small increase can easily be mitigated.



- The graph shows the overhead incurred for implementations of ASCON on Spartan-6.
- CEASAR HS Package leads to a much larger area increase than the LW Package as it expands the data and key buses to the full block size.



- CAESAR LW Package allows for bus widths of 8 and 16 bits, which are not currently supported by CAESAR HS Package.
- The CAESAR LW-Package reduces the design time for LW implementations.
- The CAESAR LW Package will be included in the next release of the *Development Package for the CAESAR Hardware API*.
- The usage will be documented in the next release of the *Implementer's Guide to the CAESAR Hardware API*.

Acknowledgment

The CAESAR Lightweight API Support Package was developed in collaboration with Fabrizio De Santis and Michael Tempelmeier from Technische Universität München

References

[1] E. Homsirikamol, W. Diehl, A. Ferozpuri, F. Farahmand, P. Yalla, J.-P. Kaps, and K. Gaj, "CAESAR hardware API," Cryptology ePrint Archive, Report 2016/626, 2016, http://eprint.iacr.org/2016/626.

[2] "Development package for the CAESAR hardware APIv1.2," https://cryptography.gmu.edu/athena/AEAD/GMU_AEAD_HW_API_v1_2.zip, accessed: 2017-06-30.

[3] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, and R. Van Keer, "CAESAR submission:Ketje v2," Submission to CAESAR (Round3), September 2016, https://competitions.cr.yp.to/round3/ketjev2.pdf.

[4] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "ASCON v1.2," Submission to CAESAR (Round3), September 2016.

[5] K. Gaj, J.-P. Kaps, V. Amirineni, M. Rogawski, E. Homsirikamol, and B. Y. Brewster, "ATHENa – automated tool for hardware evaluation: Toward fair and comprehensive benchmarking of cryptographic hardware using FPGAs," in *20th International Conference on Field Programmable Logic and Applications - FPL 2010*. IEEE, 2010, pp. 414–421, winner of the FPL Community Award.

[6] "ATHENa database of FPGA results for authenticated ciphers," https://cryptography.gmu.edu/athenadb/fpga_auth_cipher/table_view, accessed: 2017-07-30.