Enhancing CAESAR Hardware API Support for Lightweight Architectures

Panasayya Yalla[1] **Jens-Peter Kaps**[1] Fabrizio De Santis[2]
Michael Tempelmeier[2]

[1]Cryptographic Engineering Research Group (CERG)
`http://cryptography.gmu.edu`
Volgenau School of Engineering, GMU, USA
[2]Technische Universität München, Munich, Germany

DIAC 2016: Directions in Authenticated Ciphers 2016

Outline

Support for CAESAR Hardware API

- Specification of standard hardware Application Programming Interface (API)
- Implementer's Guide
- Development Package including VHDL code for high-speed implementations
  - Pre- and PostProcessors to handle the protocol
  - Universal Padding Unit

No VHDL Code for Lightweight Implementations

- Protocol handling is an additional burden
- Makes debugging more difficult

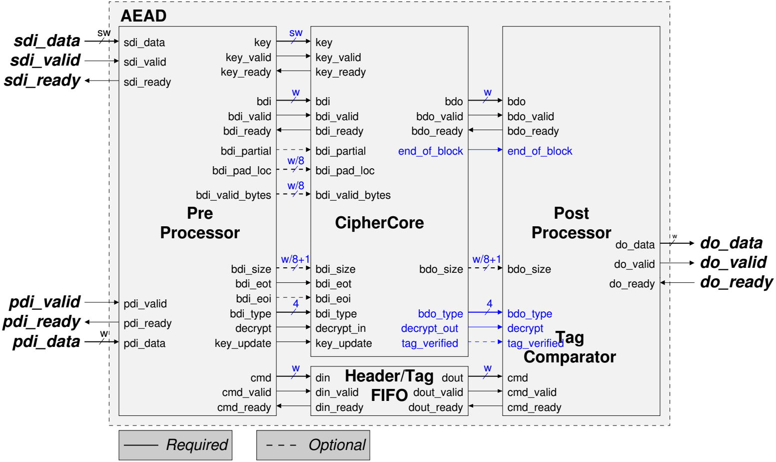
Goals for Support of Lightweight Implementations

- Reduce the burden on the designer
- Fully compliant with the CAESAR API
- Support CAESAR API's bus widths for lightweight implementations of 8, 16, and 32 bits
- Clear separation of Communication Protocol and Algorithm
- Low area footprint:
  - Avoid duplication of elements between protocol handling and algorithm
- Minimize overhead:
  - Not a 'one-size-fits-all' solution ⇒ needs tweaking
  - No universal padding unit ⇒ designer can choose most efficient way to implement

Main Components

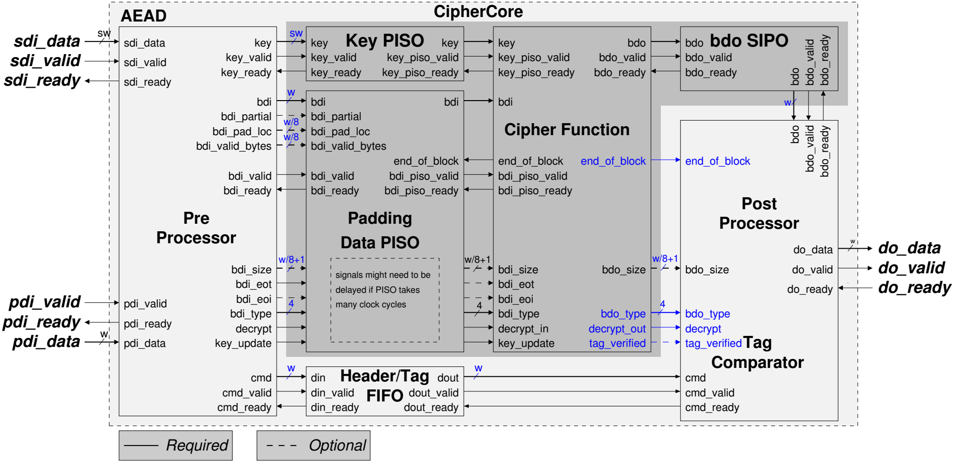
- Lightweight PreProcessor:
  - Handles protocol
  - Provides signals needed for padding
  - Separate state machines for *sdi* and *pdi*
  - 16 bits counters for segment lengths
- CipherCore (not provided):
  - Cipher function
  - Padding
  - PISO and SIPO if I/O width different than CAESAR API
- By-Pass FIFO: Stores and passes-on header information and Tag to PostProcessor
- PostProcessor
  - 16 bits counter for output segment length
  - Tag verification module

CAESAR Lightweight Block Diagram



Differences to HS are shown in blue.

CAESAR LW Block Diagram with PISO, and SIPO

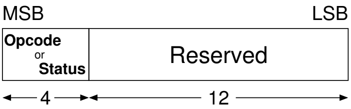


Differences between High-Speed vs Lightweight Packages

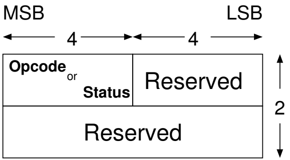
Lightweight Development Package

- Supports bus widths 8, 16, and 32 bits
- Assumes that Padding is performed in *CipherCore*
- Sets the width of all data buses between modules *PreProcessor, CipherCore, FIFO, and PostProcessor* equal to bus width **w**
- Moves the Tag comparison to the PostProcessor (when possible)
- Does not provide data storage other than for control signals in the *PreProcessor*

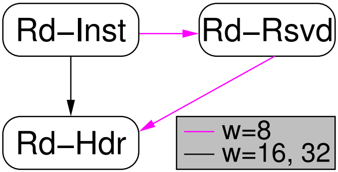
Protocol: Instruction



16-bit Instruction with w=16



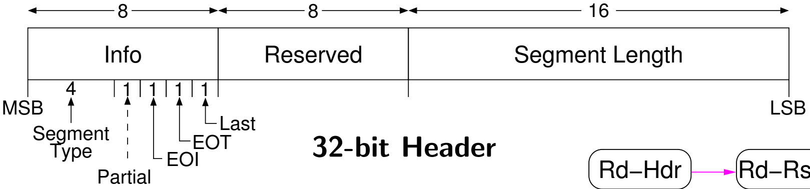
16-bit Instruction with w=8



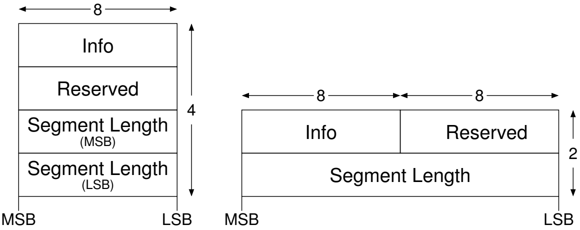
States for Processing Instruction

Opcode	Description	Status	Description
0010	Authenticated Encryption (ENC)	1110	Success
0011	Authenticated Decryption (DEC)	1111	Failure
0100	Load Key (LDKEY)	Others	Reserved
0111	Activate Key (ACTKEY)		

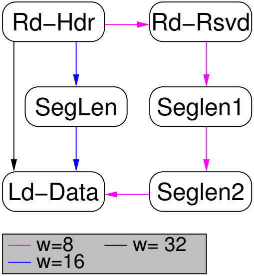
Protocol: Segment Header



32-bit Header



With w= 8, and 16



States for Processing Header

Case Study

- Implementation of Ketje-Sr with integrated support of CAESAR API
- Implementation of Ketje-Sr using new CAESAR lightweight development package
  $\Rightarrow$ Determine overhead of CAESAR LW package
- Implementation of ASCON using CAESAR LW package
  $\Rightarrow$ Using CAESAR LW package on "unknown" algorithm

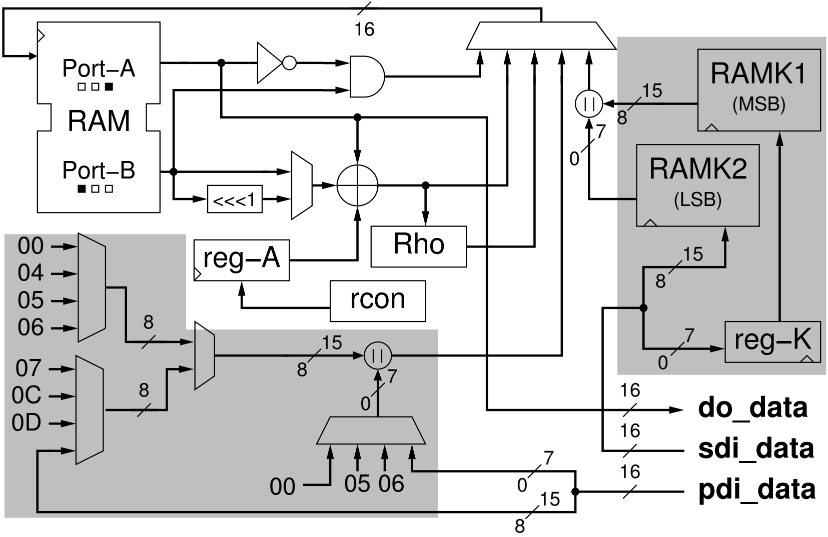
Taget Devices for Benchmarking

Vendor	FPGA Family	Device	Tool	Optimization
Xilinx	Spartan6 Artix7	xc6slx16csg324-3 xc7a100tcsg324-3	Xilinx ISE 14.7	ATHENa
Altera	Cyclone IV Cyclone V	ep4ce22f17c6 5ceba4f23c7	Quartus Prime 16.0.0	

Ketje

- Ketje 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

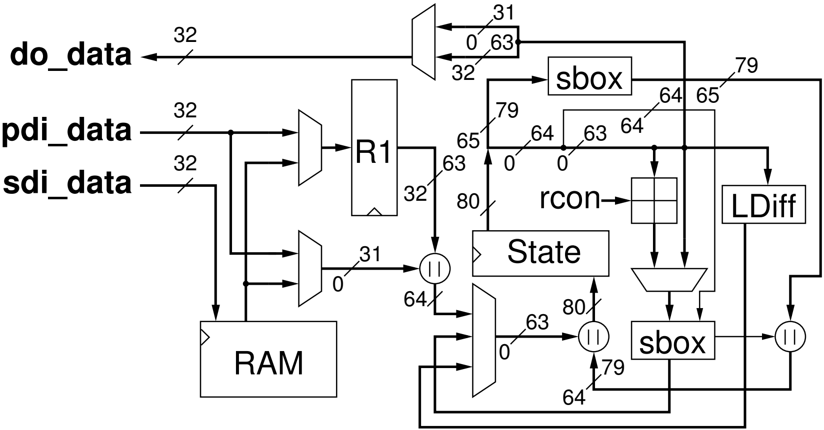


- We implemented a Ketje-Sr using a 16-bit datapath and interface.
- Datapath is the same for integrated CAESAR API support and using CAESAR LW package.
- State is stored in a dual-port memory (RAM) with one read/write and one read-only ports.
- To reduce the complexity of padding for key, the key size 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.
- $TP = \frac{32}{160} * F$

Ascon

- Ascon 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
- 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 80-bit datapath is used in this design and a 32-bit interface.
- The state is stored in a shift-register, which either shifts by 64 bits or 80 bits.
- 80-bit shifts are used for substitution operation as 64 is not a multiple of 5 (5x5 S-boxes).
- 5x5 S-Boxes are implemented using look-up tables.
- The round constants are generated using two 4 bit registers and adders.
- Key is stored in a RAM
- Two 5-to-1 multiplexers are used to perform circular shifts in linear diffusion step (LDiff)
- $TP = \frac{32}{54} * F$

Implementation Results on Xilinx Spartan 6 FPGA

Design	Slices	LUTs	FFs	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[2]	203	645	393	137.5	154.25	0.76
Joltik[3]	168	534	381	200.0	426.67	2.54
ASCON[4]	202	540	383	231.0	1,852.80	9.17

1 ⇒ Integrated CAESAR API; 2 ⇒ CAESAR LW Package;
3 ⇒ No compliance with CAESAR API (non-GMU design);
4 ⇒ CAESAR HS Package (tweaked, non-GMU design);

- 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.

Area Overhead HS vs. LW Packages

Design	API	Slices	LUTs	FFs
LW ASCON	Yes	203	645	393
	No	173	619	357
Overhead		30	26	35
HS ASCON	Yes	771	2,299	832
	No	720	2,264	484
Overhead		51	35	348

- Results for ASCON with w=32 on Spartan 6.
- Supporting CAESAR API using LW Package leads to a small area increase.
- Supporting CAESAR API using HS Package leads to a larger area increase.

Conclusion

- CAESAR LW Package allows for bus widths of 8 and 16 bits, which are not currently supported by CAESAR HS Package.
- Using CAESAR LW Package leads to a small area increase over integrated designs, however this can be easily mitigated.
- CEASAR HS Package leads to a much larger area increase than the LW Package as it converts between the word width and block width data buses.
- 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*.