A Universal Hardware API for Authenticated Ciphers

Ekawat Homsirikamol,
William Diehl, Ahmed Ferozpuri,
Farnoud Farahmand,
Malik Umar Sharif, and Kris Gaj
George Mason University
USA

http://cryptography.gmu.edu
https://cryptography.gmu.edu/athena

Co-Authors from the GMU CAESAR Team

"Ice"
Homsirikamol

Will
Diehl

Farnoud
Farahmand

Umar
Sharif

Kris
Gaj

CAESAR Contest 2013-2017

Cryptographic Transformations

Secret-Key Ciphers

Block Ciphers **Stream Ciphers**

confidentiality

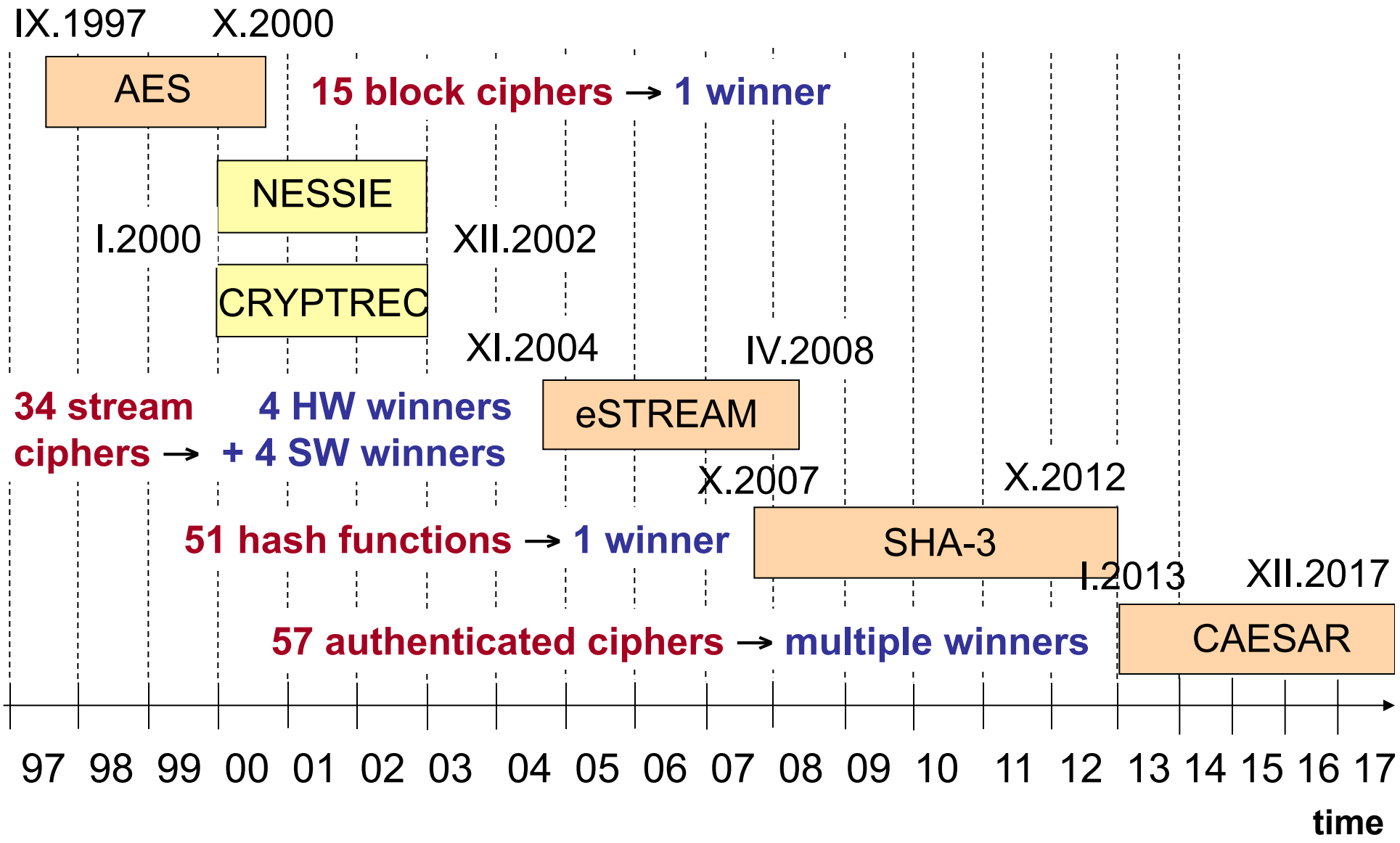
MACs
&
Hash Functions

message integrity & authentication

Authenticated Ciphers

confidentiality + message integrity & authentication

Cryptographic Standard Contests



CAESAR Contest Timeline

- 2014.03.15: Deadline for **first-round submissions**
- 2014.04.15: Deadline for **first-round software**
- 2015.07.07: Announcement of **second-round candidates**
- 2015.09.15: Deadline for **second-round software**

- 2015.12.15: Deadline for **second-round Verilog/VHDL**
- 2016.03.15: Announcement of **third-round candidates**
- 2016.12.15: Announcement of **finalists**
- 2017.12.15: Announcement of **final portfolio**

Evaluation Criteria

Security

Software Efficiency

µProcessors µControllers

Hardware Efficiency

FPGAs ASICs

Flexibility

Simplicity

Licensing

Software API vs. Hardware API - Similarities

Software API	Hardware API
Function Arguments	Core Ports = Interface
Argument Format	Input/Output Format

Software API vs. Hardware API - Differences

Software API	Hardware API
Abstracted	Physically constrained
All inputs available at the same time	Inputs accepted, outputs produced sequentially

Hardware API and the CAESAR Contest

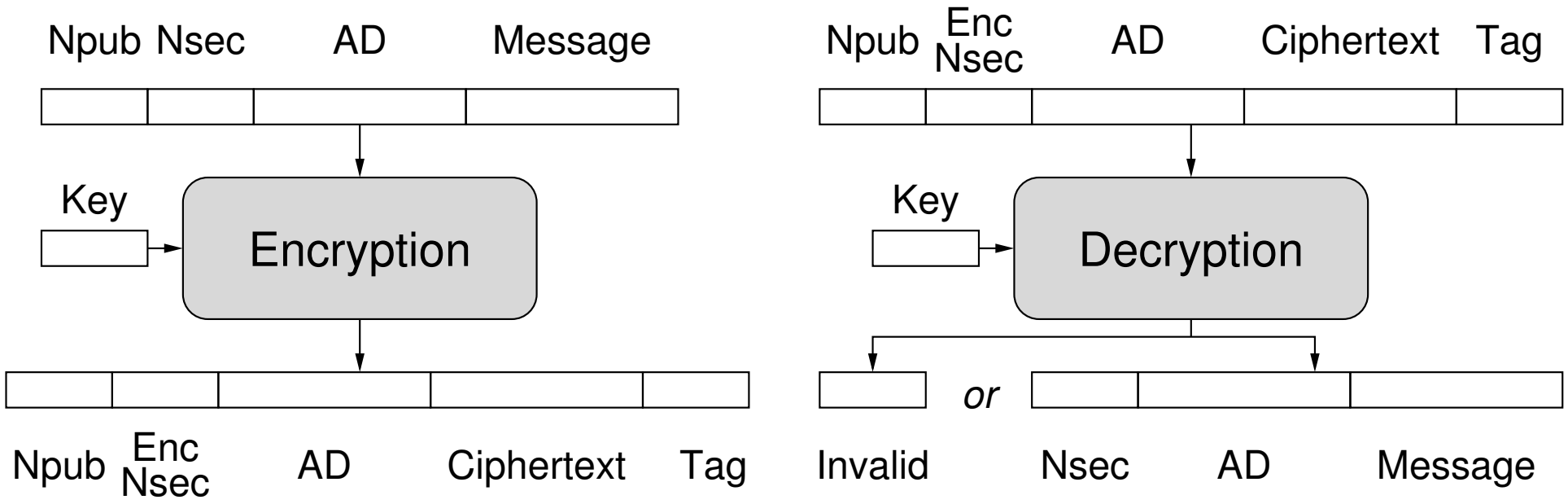
- Hardware API can have a high influence on Area and Throughput/Area ratio of all implementations
- Hardware API typically much more difficult to modify than Software API
- Without a comprehensive hardware API, the comparison of existing and future implementations highly unreliable and potentially unfair
- Need for a uniform hardware API, endorsed by the CAESAR Committee, and adopted by all future implementers

Previous Work

- **Popular general-purpose interfaces**
  - **ARM:** **AXI4, AXI4-Lite, AXI4-Stream** (Advanced eXtensible Interface)
  - **IBM:** **PLB** (Processor Local Bus), **OPB** (On-chip Peripheral Bus)
  - **Altera: Avalon**
  - **Xilinx: FSL** (Fast Simplex Link)
  - **Silicore Corp.: Wishbone** (used by opencores.org)
- **Hardware APIs used during the SHA-3 Contest**
  - **GMU, Virginia Tech, University College Cork, etc.**
- **Interfaces used so far in the CAESAR competition**
  - **minimalistic, algorithm specific**
  - **AXI4-Stream-based proposed by ETH (non-uniform, algorithm-specific user and data ports)**


Proposed API
Specification

Input and Output of an Authenticated Cipher



Npub (Public Message Number), typically Nonce

Nsec (Secret Message Number) [supported by few algorithms]

Enc Nsec – Encrypted Secret Message Number

AD – Associated Data

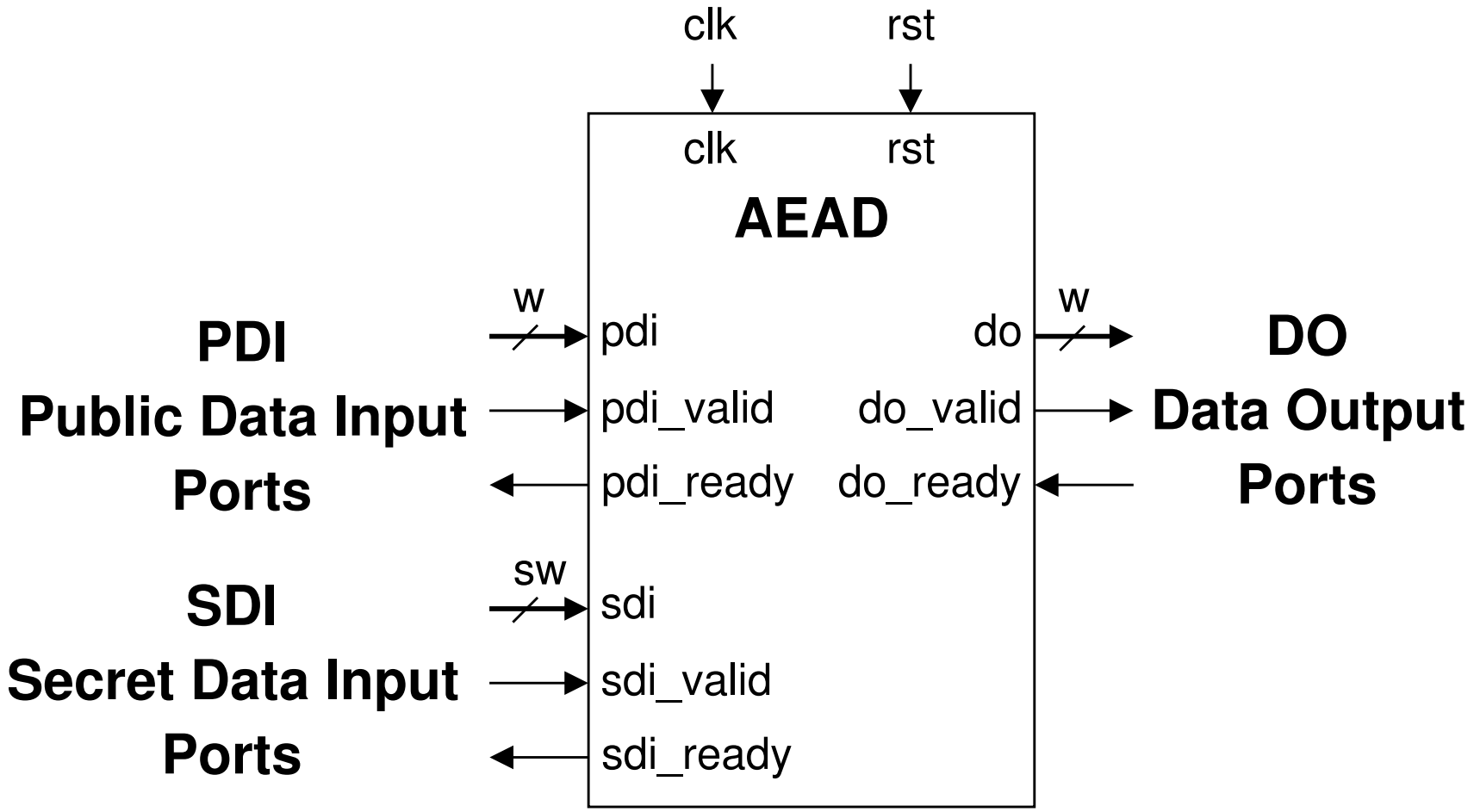
Proposed Features (1)

- inputs of arbitrary size in bytes (but a multiple of a byte only)
- size of the entire message/ciphertext does not need to be known before the encryption/decryption starts (unless required by the algorithm itself)
- independent data and key inputs
- wide range of data port widths, $8 \leq w \leq 256$
- simple high-level communication protocol
- support for the burst mode
- possible overlap among processing the current input block, reading the next input block, and storing the previous output block

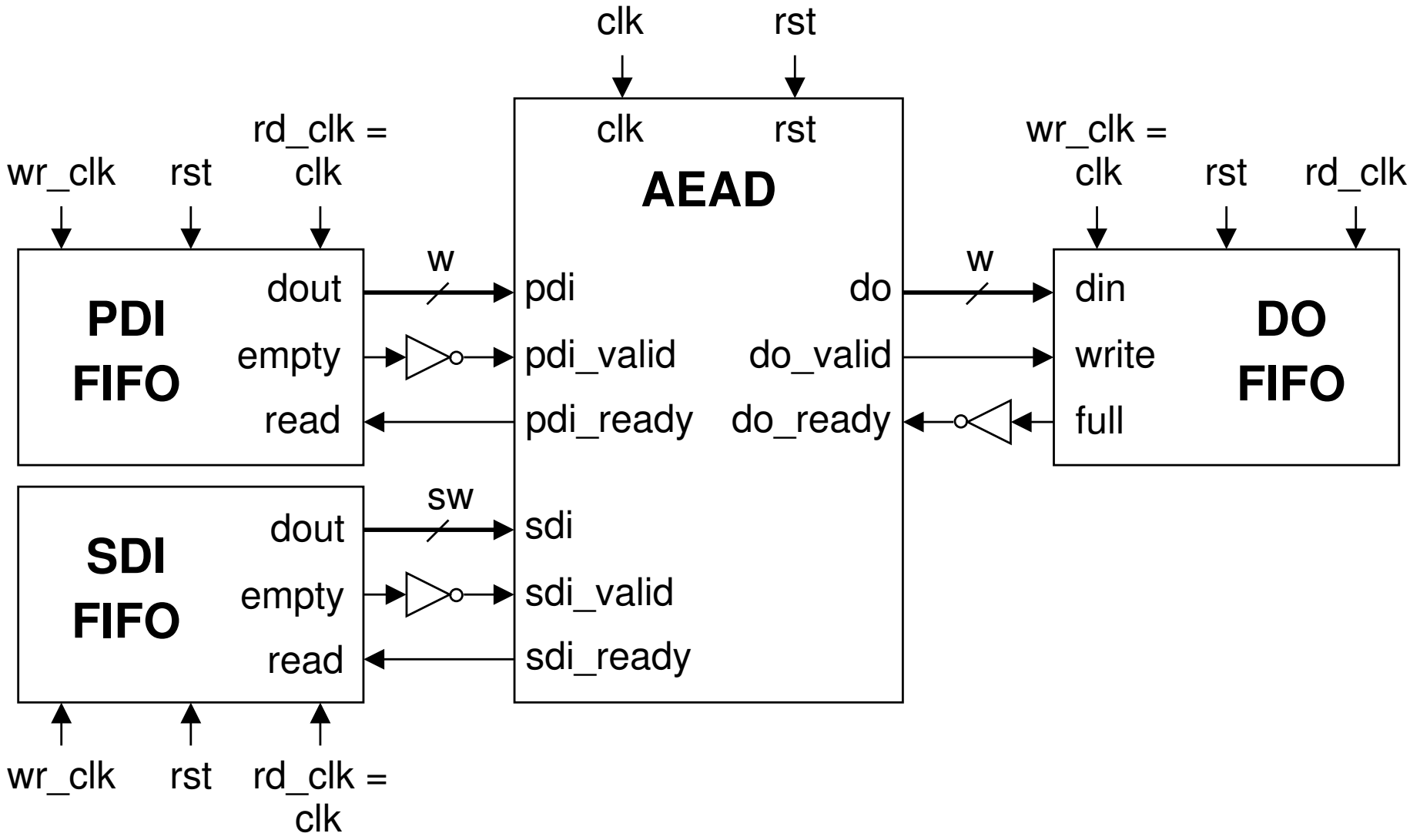
Proposed Features (2)

- storing decrypted messages internally, until the result of authentication is known
- support for encryption and decryption within the same core, but only one of these two operations performed at a time
- ability to communicate with very simple, passive devices, such as FIFOs
- ease of extension to support existing communication interfaces and protocols, such as
  - AMBA-AXI4 from ARM - a de-facto standard for the System-on-Chip buses
  - PCI Express – high-bandwidth serial communication between PCs and hardware accelerator boards

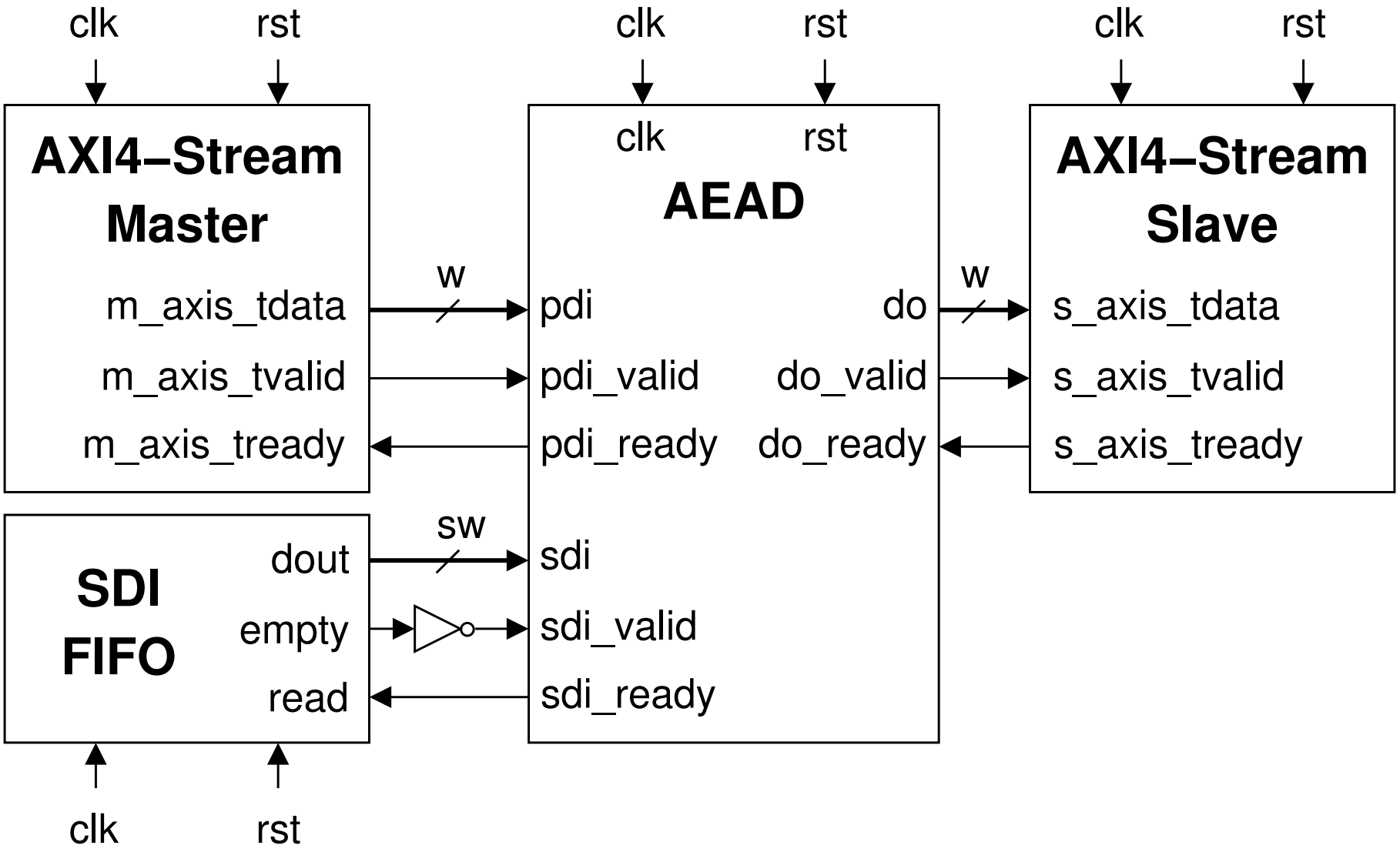
Authenticated Encryption with Associated Data – AEAD Interface



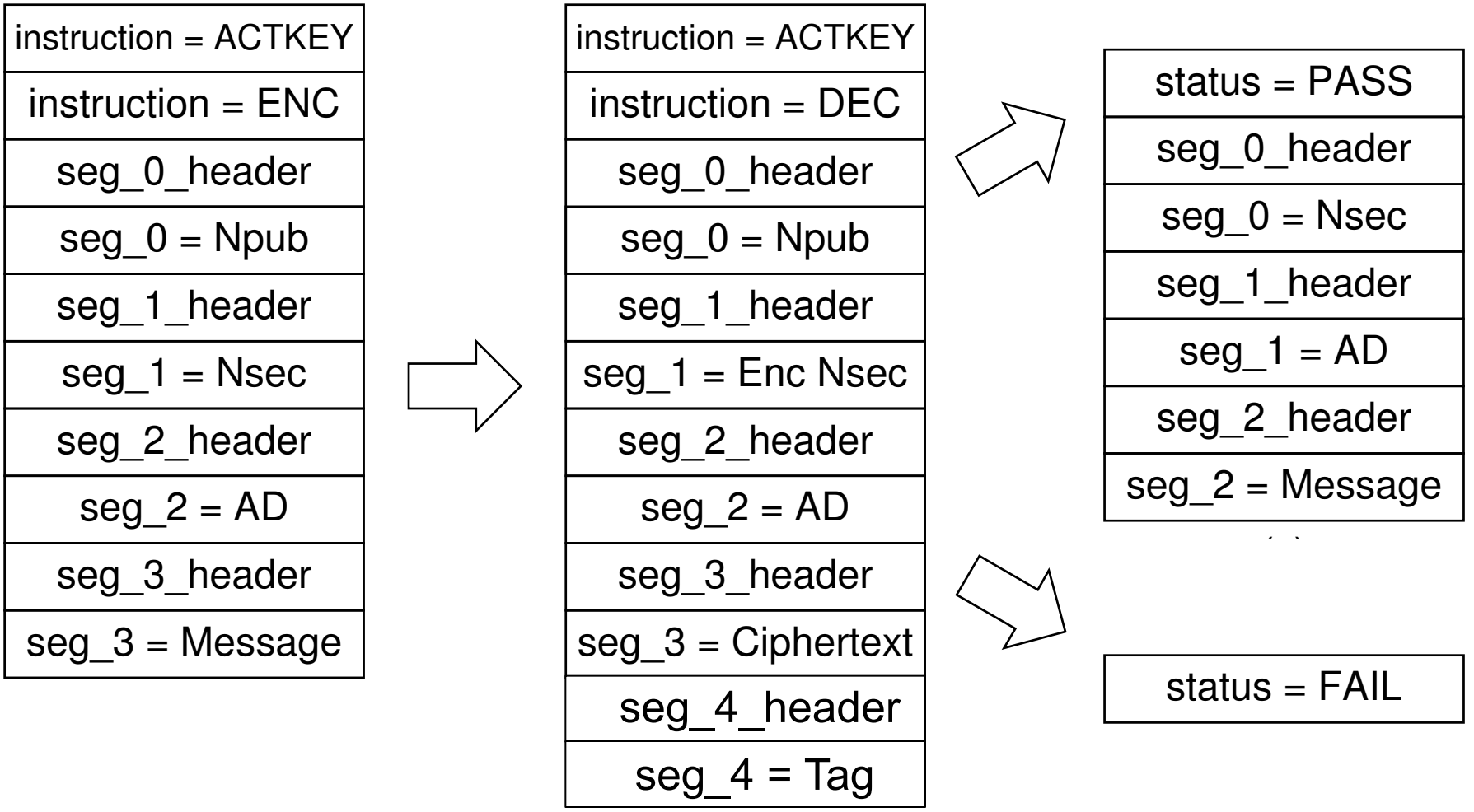
Typical External Circuits (1) - FIFOs



Typical External Circuits (2) – AXI4 IPs



Input/Output Format



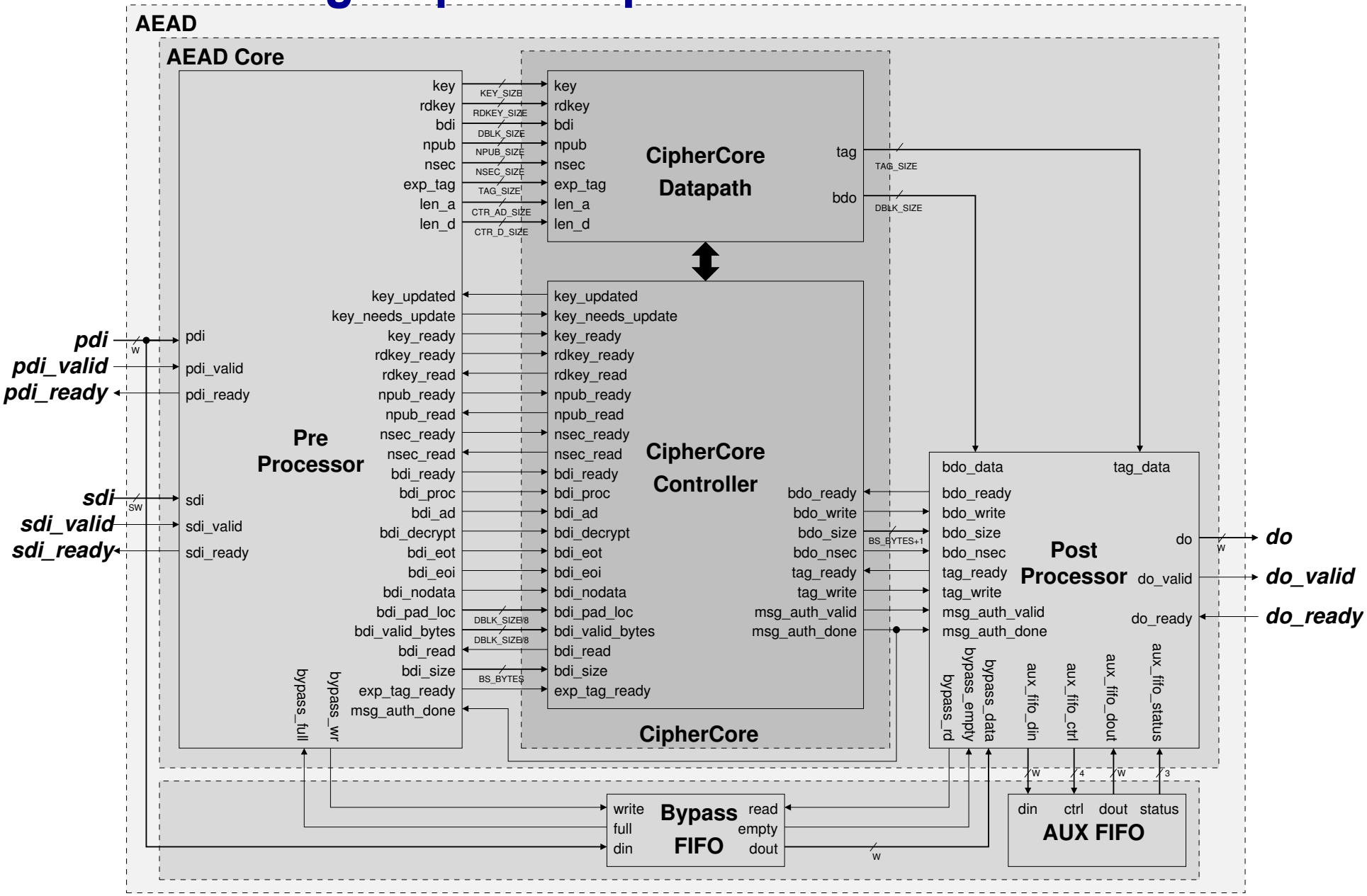
PDI for encryption

DO for encryption = PDI for decryption

DO for decryption

High-Speed Implementations of AEAD

Block Diagram of a High-Speed Implementation of AEAD



PreProcessor and PostProcessor for High-Speed Implementations (1)

PreProcessor:

- parsing segment headers
- loading and activating keys
- Serial-In-Parallel-Out loading of input blocks
- padding input blocks
- keeping track of the number of data bytes left to process

PostProcessor:

- clearing any portions of output blocks not belonging to ciphertext or plaintext
- Parallel-In-Serial-Out conversion of output blocks into words
- formatting output words into segments
- storing decrypted messages in AUX FIFO, until the result of authentication is known
- generating an error word if authentication fails

PreProcessor and PostProcessor for High-Speed Implementations (2)

Features:

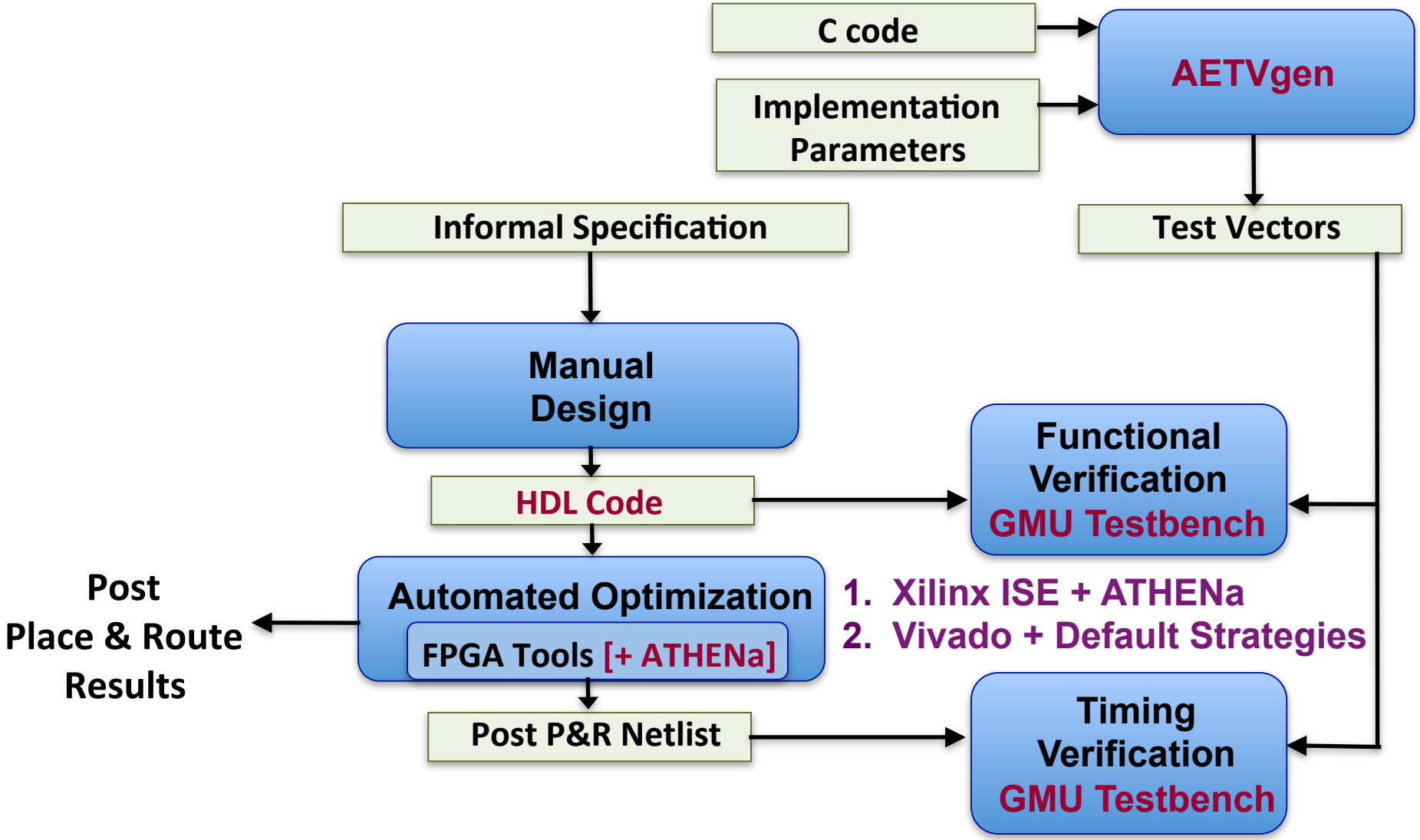
- Ease of use
- No influence on the maximum clock frequency of AEAD (up to 300 MHz in Virtex 7)
- Limited area overhead
- Clear separation between the AEAD Core unit and internal FIFOs
  - Bypass FIFO – for passing headers and associated data directly to PostProcessor
  - AUX FIFO – for temporarily storing unauthenticated messages after decryption

Benefits:

- The designers can focus on designing the CipherCore specific to a given algorithm, without worrying about the functionality common for multiple algorithms
- Full-block width interface of the CipherCore


Verification
& Result
Generation

Proposed Development & Benchmarking Flow



Universal Testbench & Automated Test Vector Generation

- Universal Testbench supporting any authenticated cipher core following GMU AEAD API
- Change of cipher requires only changing test vector file
- A Python script, AETVgen, created to automatically generate test vector files representing multiple test cases
  - Encryption and Decryption
  - Empty Associated Data and/or Empty Message/Ciphertext
  - Various, randomly selected sizes of AD and Message/Ciphertext
  - Valid tag and invalid tag cases
- All source codes made available at GMU ATHENa website

Result Generation

- Generation of results possible for
  - CipherCore – full block width interface, incomplete functionality
  - AEAD Core - recommended
  - AEAD – difficulty with setting BRAM usage to 0 (if desired)
- Use of wrappers:
  - Out-of-context (OOC) mode available in Xilinx Vivado (no pin limit)
  - Generic wrappers available in case the number of port bits exceeds the total number of user pins, when using Xilinx ISE
  - GMU Wrappers: 5 ports only (clk, rst, sin, sout, piso_mux_sel)
- Recommended Automated Optimization Procedure
  - ATHENa for Xilinx ISE and Altera Quartus II
  - 25 default optimization strategies for Xilinx Vivado

Authenticated Ciphers Implemented Using Our Hardware API

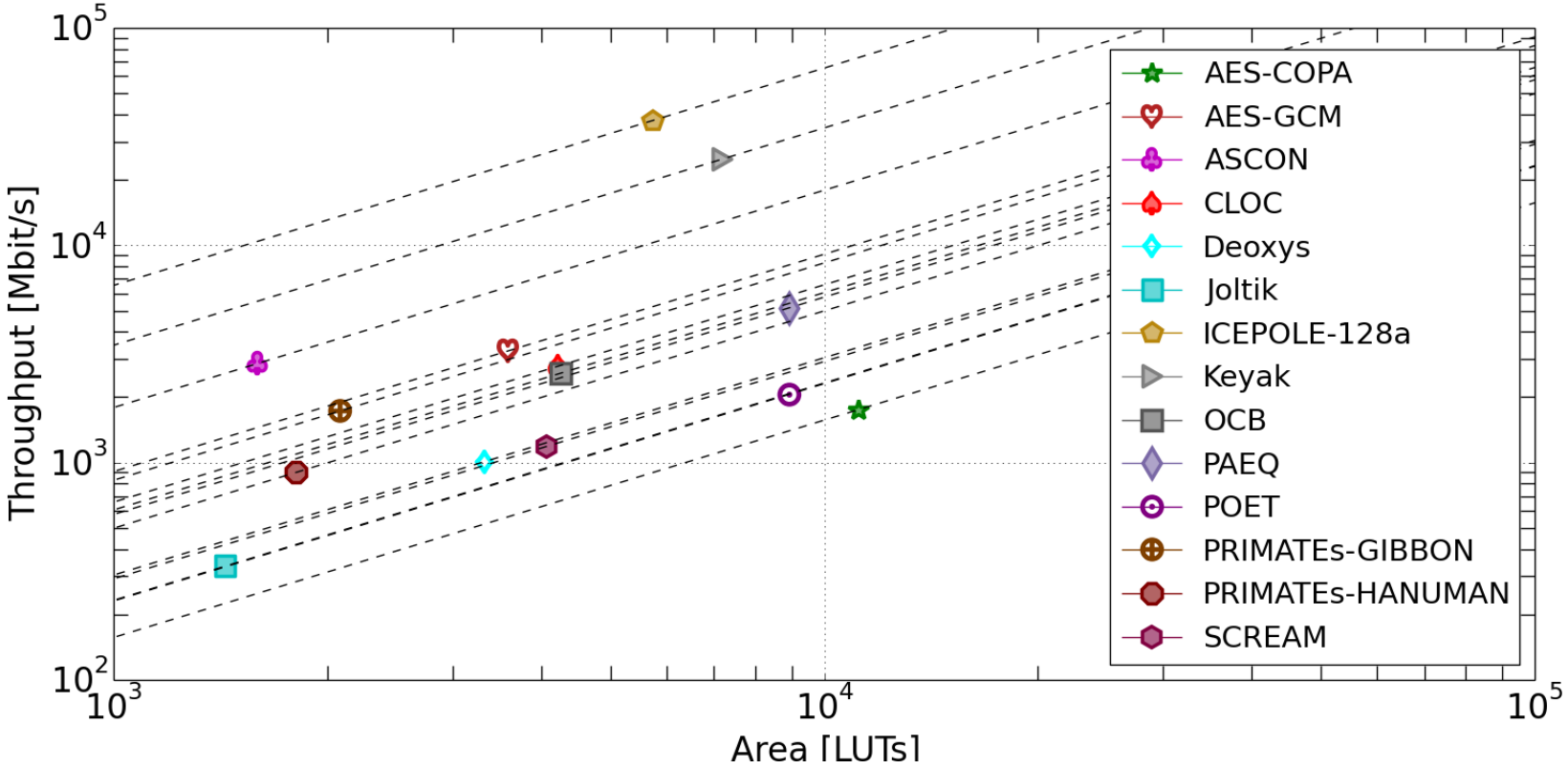
CAESAR Candidates:

1. AES-COPA
2. Ascon
3. CLOC
4. Deoxys
5. ICEPOLE
6. Joltik
7. Keyak
8. Minalpher
9. OCB
10. PAEQ
11. POET
12. PRIMATEs-HANUMAN
13. PRIMATES-GIBBON
14. SCREAM

Current Standard:

15. AES-GCM

Preliminary Results for 13 Round 1 CAESAR Candidates & AES-GCM



ATHENa Database of Results for Authenticated Ciphers

http://cryptography.gmu.edu/athena

ATHENA

AUTOMATED TOOL FOR HARDWARE EVALUATION

About

All FPGA Results

FPGA Rankings

Login

Authenticated Encryption FPGA Ranking

Show Help

Result Filtering

Algorithm Group

- ○ Round 2 CAESAR Candidates and current standards
- ◉ Round 1 CAESAR Candidates and current standards

Implementation Type:

- ◉ High Speed Implementations, Single Message Architectures
- ○ High Speed Implementations, All Architectures
- ○ Low Area Implementations

Implementation Approach:

- ○ Register Transfer Level
- ○ High Level Synthesis
- ○ HW/SW Codesign
- ◉ Any

Hardware API:

- ☑ GMU_AEAD_Core_API_v1
- ☐ GMU_AEAD_API_v1
- ☐ GMU_CipherCore_API_v1
- ☐ Full-Block width(custom)
- ☐ GMU_AEAD_Core_API_v0

Key Size:

- ○ 128
- ○ From 80 To 256
- ◉ Any

ATHENa Database of Results for Authenticated Ciphers

Throughput for:

- (•) Authenticated Encryption
- ( ) Authenticated Decryption
- ( ) Authentication Only

Min Area: 0
Max Area: 1000000
Min Throughput: 0
Max Throughput: 1000000

Source:

- [ ] Source Available

Ranking:

- (•) Throughput/Area
- ( ) Throughput
- ( ) Area

Please note that codes with primitives, megafunctions, or embedded resources are not fully portable.

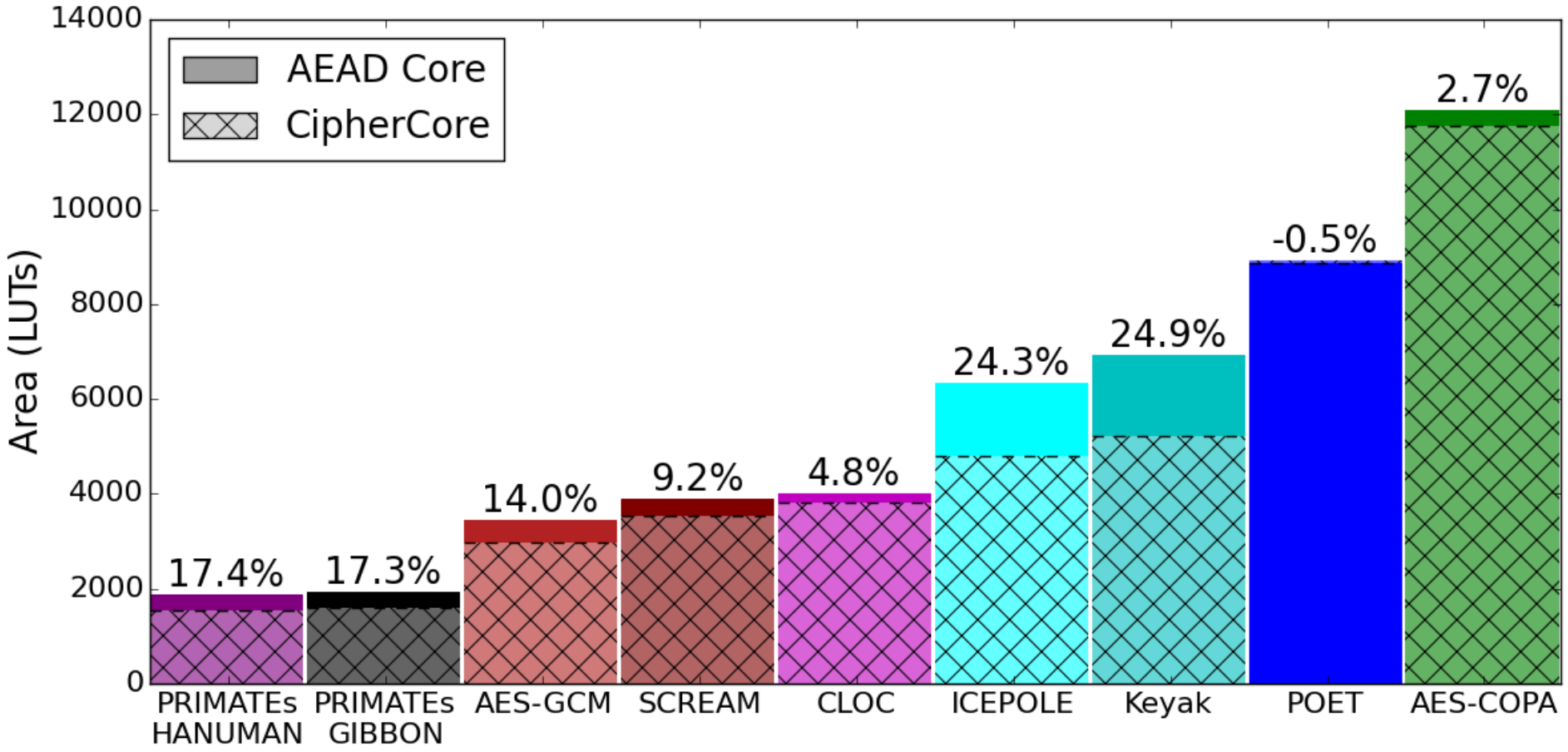
Update

Compare Selected

Show 25 entries

Result ID	Algorithm (Disable Unique)	Key Size [bits]	Implementation Approach	Hardware API	Arch Type
154	ICEPOLE	128	RTL	GMU_AEAD_Core_API_v1.1	Basic Iterative
73	Keyak	128	RTL	GMU_AEAD_Core_API_v1	Basic Iterative
62	AES-GCM	128	RTL	GMU_AEAD_Core_API_v1	Basic Iterative
65	CLOC	128	HLS	GMU_AEAD_Core_API_v1	Basic Iterative
80	PRIMATEs-GIBBON	120	RTL	GMU_AEAD_Core_API_v1	Basic Iterative
144	OCB	128	RTL	GMU_AEAD_Core_API_v1	Basic Iterative
124	PRIMATEs-HANUMAN	120	HLS	GMU_AEAD_Core_API_v1	Basic Iterative
86	SCREAM	128	RTL	GMU_AEAD_Core_API_v1	Basic Iterative
142	Joltik	128	RTL	GMU_AEAD_Core_API_v1	Basic Iterative
75	POET	128	RTL	GMU_AEAD_Core_API_v1	Basic Iterative
60	AES-COPA	128	RTL	GMU_AEAD_Core_API_v1	Basic Iterative

AEAD Core vs. CipherCore Area Overhead for Virtex-6



$$\text{Overhead} = \frac{\text{LUT(AEAD\_Core)-LUT(CipherCore)}}{\text{LUT(AEAD\_Core)}} \times 100\%$$

Conclusions

- Complete Hardware API for authenticated ciphers
  - The Detailed Specification - ePrint 2015/669
- GMU hardware API with supporting materials
  - Available at: https://cryptography.gmu.edu/athena
  - Universal testbench and Automated Test Vector Generation
  - PreProcessor and PostProcessor Units for high-speed implementations
  - Universal wrappers and scripts for generating results
  - Ease of recording and comparing results using ATHENa database
- GMU proposal open for discussion and possible improvements through
  - Better specification
  - Better implementation of supporting codes

Future Work

- Support for two-pass algorithms
- Pipelining of multiple streams of data
- Detection and reporting of input formatting errors
- Accepting inputs with padding done in software

Thank you!

Comments?

Questions?

Suggestions?

http://cryptography.gmu.edu/athena
Choose: Download
http://cryptography.gmu.edu