Verilog/ VHDL Code:
Suggested List of Deliverables

I. Introduction

In order to simplify benchmarking and any further optimizations of the submitted Verilog/VHDL implementations, we propose a uniform structure of all deliverables.

All implementations that share the same source code, except of different values of generics and/or constants, should be submitted as a single .zip file.

In order to prepare this .zip file, please follow the instructions given below.

Please prepare the top-level folder called

<Authenticated_Cipher_Name>_<Implementation_Team_Name>

e.g., ASCON_GMU.

Within this folder, please create the following structure of second-level folders:

```
|-docs
|-src_rtl
|-src_tb
|-scripts
|-KAT
|-bd
|-results
```

The recommended content of these folders is described below:

II. List of Deliverables

1. Assumptions

File: `docs/assumptions.pdf` or `docs/assumptions.txt`
(depending on the file format used)

This file should contain at least the following information:

A. Hardware description language used

VHDL, Verilog, or Mixed (VHDL and Verilog).

B. Type of implementation

High-speed or lightweight.

C. Use of hardware description language source files provided as a part of the Development Package

Please include the following table:

File name	Used (Y/N)	Release number*	Functional modifications** (Y/N)
PreProcessor.vhd			
PostProcessor.vhd			
fwft_fifo.vhd			

* The Release number refers to a version of the Development Package for the CAESAR Hardware API (e.g., v1.0-1, v1.0-3, etc.)

** Functional modifications refer to any changes other than the changes related to the list and default values of generics.

D. Supported types and order of segment types

Please list an order of segment types supported by your implementation, using the following abbreviations:

npub	: public message number
nsec	: secret message number
ensec	: encrypted secret message number
ad	: associated data
ad_npub	: associated data || npub
npub_ad	: npub || associated data
data	: data (plaintext/ciphertext)
data_tag	: data (plaintext/ciphertext) || tag
tag	: tag

Please note that according to the CAESAR Hardware API:

- ad, ad_npub, npub_ad, data, and data_tag can be divided into multiple segments of the same type (each limited to maximum of 2^{16}-1 bytes for single-pass algorithms, and 2^{11}-1 for two-pass algorithms)
- npub, nsec, ensec, and tag are always composed of only one segment.

For clarity, please provide the order of segment types for all four cases:

a. input to encryption, e.g.: nsec npub_ad data
b. output from encryption, e.g.: ensec data_tag
c. input to decryption, e.g.: ensec npub_ad, data_tag
d. output from decryption, e.g.: nsec data

Please note that all of the above orders can be expressed using the following single option of the aeadtvgen app:

```
--msg_format nsec npub_ad data_tag
```

E. Deviations from the CAESAR Hardware API v1.0 specification

These deviations may include deviations regarding the following components of the API:

B.1 Minimum compliance criteria

Please list all deviations from the criteria described in Section 1 of the CAESAR API specification.

For example:

- the core supports only encryption,
- the core handles only associated data, messages, and ciphertexts composed of full blocks
- AD and/or message is assumed to be padded before entering the AEAD core
- unused portions of the last block are not cleared before being sent to the output port do
- the AEAD core does not support empty AD and/or an empty message
- the supported maximum sizes of AD/plaintext/ciphertext are smaller than the limits described in the API specification and its appendix (2^{32}-1 for single-pass algorithms, and 2^{11}-1 for two-pass algorithms)
- the core requires two or more clocks (with different frequencies)
- the widths of the PDI , DO, or SDI data ports do not match the requirements described in the specification for a given type of implementation (high-speed vs. lightweight).

B.2 Interface

Please list all deviations from the interface described in Section 2 of the CAESAR API specification.

For example:

- any differences in the names, widths, and/or meanings of ports
- different width of pdi_data and do_data, etc.

B.3 Protocol

Please list all deviations from the protocol described in Section 3 of the CAESAR API specification.

For example:

- no support for multiple consecutive segments of the type: AD, Plaintext, and Ciphertext (or Ciphertext||Tag if appropriate)
- special use for Reserved fields of an Instruction/Status or a Segment Header
- extra words added beyond the minimum number of words necessary to input AD, message, ciphertext, etc. of a given length (e.g., to always enter data in full block chunks)

- extra zeros added in the input words other than the last words of a given type (e.g., to handle the case when the data port width, w, is not a divisor of the data block size).

B.4 Timing characteristics

Please list all deviations from the timing characteristics described in Section 4 of the CAESAR API specification.

For example:

- a different order of bytes within a word of data bus.

F. Disagreement with the Appendix to the CAESAR Hardware API v1.0

Please declare any disagreement with the Appendix to the CAESAR Hardware API.

For example:

- the use of a Length segment as a required input to an "online" algorithm, such as AES-GCM, in which all lengths can be calculated as the AD/plaintext/ciphertext arrives and is processed
- a different format of the Length segment in an “offline” algorithm, such as AES-CCM, understood as an algorithm that require the availability of the lengths of the AD and plaintext (ciphertext) in advance, before the authenticated encryption (decryption) starts.
- no use of an external FIFO in the implementation of a two-pass algorithm
- no support for the generic G_MAX_LEN in the implementation of a *single-pass* algorithm. This generic should allow the choice between two maximum lengths of AD/plaintext/ciphertext:
    - Maximum length for single-pass algorithms: 2^{32}-1
    - Maximum length for two-pass algorithms: 2^{11}-1.

2. Variants

File: `docs/variants.pdf` or `docs/variants.txt`
(depending on the file format used)

We define variants of the design as different versions of the design that

A. share the same synthesizable source code
B. share the same testbench
C. differ only with values of generics or constants.

Different variants may correspond to

- different algorithms of the same family
- different sizes of keys, nonces, tags, etc.
- different parameters of the interface, such as `w` and `sw`
- variants with and without a secret message number, Nsec
- different hardware architectures (e.g., basic iterative, unrolled, folded, pipelined, etc.)

Please describe in this file all variants recommended for hardware benchmarking in the order of your preference (primary recommendations first).

For each variant, provide at least the following information:

a. unique identifier, e.g., v1, v2
b. name of the variant (optional)
c. name of the corresponding reference software implementation (optional)
d. all non-default values of generics and constants
e. formulas for the
    - key setup time (in clock cycles)
    - execution time of authenticated encryption (in clock cycles), as a function of the number of associated data blocks, Na, and the number of message blocks, Nm (excluding any key setup cycles)
    - execution time of authenticated decryption (in clock cycles) as a function of the number of associated data blocks, Na, and the number of ciphertext blocks, Nc (excluding any key setup cycles)

All formulas should be confirmed using functional simulation.

Please do your best to limit the number of variants recommended for hardware benchmarking (e.g., by including only primary variants of the CAESAR algorithms declared in the algorithm specification, and/or by performing initial design space exploration using FPGA tools).

3. Synthesizable source code

Folder: `src_rtl`

Please place in this folder all synthesizable source files, including any files being a part of the Development Package for the CAESAR Hardware API (such as AEAD.vhd. AEAD_Arch.vhd, PreProcessor.vhd, PostProcessor.vhd, etc.).

Please make sure to set the default values of generics in the top-level file (such as AEAD.vhd) and the default values of constants in the corresponding package (such as AEAD_pkg.vhd) to values specific to the primary variant of your algorithm.

Please also place in the same folder the file `source_list.txt`, containing the list of all design files in the bottom-up order, i.e., packages and low-level units first, and the top-level unit last.

4. Testbench

Folder: `src_tb`

Please place in this folder only your testbench and any non-synthesizable source files used by your testbench.

In case you use the universal testbench provided as a part of the Development Package, these files should include only `AEAD_TB.vhd` and `std_logic_1164_additions.vhd`.

Please also place in the same folder the file `source_list.txt`, containing the list of all testbench files in the bottom-up order, i.e., packages and low-level units first, and the top-level unit last.

5. Simulation scripts (optional)

Folder: `scripts`

Place in this folder all simulation scripts, such as `modelsim.tcl`.

6. Known-answer tests

Folder: `KAT`

Create subfolders, named `v1`, `v2`, `v3`, etc., corresponding to unique identifiers of variants, defined using recommendations described in Section 2 Variants.

In each respective subfolder, place test vector files you have used to verify your implementation of a particular variant.

It is recommended that all test vectors are described using two formats:

A. format accepted by the universal testbench `AEAD_TB.vhd` (including the `pdi.txt`, `sdi.txt`, and `do.txt files`), generated *by default* by the aeadtvgen program, and
B. a simplified format, listing each input and expected output component (e.g., key, npub, ad, pt, ct, tag) using a sequence of hexadecimal digits located in the same line, e.g.

```
key     = 55565758595A5B5C5D5E5F6061626364
npub    = B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF
ad      =
pt      = FF
ct      = 76
tag     = FDB8FCCD8A5C78DC9445457B341F13B2
```

All test vectors should be placed in the same file `test_vectors.txt`, separated by at least one empty line. This file can be automatically generated by the aeadtvgen app by using the option `--human_readable`.

Please include in the aforementioned files `pdi.txt`, `sdi.txt`, `do.txt`, and `test_vectors.txt` only test vectors that successfully passed verification.
Place all test vectors that did not pass verification in separate files:
`pdi_failed.txt`, `sdi_failed.txt`, `do_failed.txt`, and `test_vectors_failed.txt`.

7. Block diagrams (optional)

Folder: `bd`

If possible, please include a simplified block diagram of the datapath for the primary variant of your algorithm. For consistency, and future use in publications, please consider

using *Rules for Reduced Complexity Block Diagrams*, developed by William Diehl from GMU, and made available at
https://cryptography.gmu.edu/athena/CAESAR_HW_API/Reduced_Complexity_Block_Diagrams.pdf

8. License (optional)

File: `LICENSE.txt`

Include in this file any licensing and copyright information that applies to your code.

9. Preliminary results (optional)

Folders: `results/fpga` or `results/asic` (depending on technology)

The GMU Team is planning to perform hardware benchmarking of all Round 2 candidates for FPGA technology only, using approach described in the Implementer's Guide to the CAESAR Hardware API, Section 8, Generation and Publication of Results.

In order to allow the comparison of designs in terms of Resource Utilization, the GMU implementation runs will enforce the use of no DSP units and no embedded block memories.

Each team is encouraged to produce and include in their submission the preliminary results of their own benchmarking runs, conducted using a similar approach (possibly without ATHENa and Vivado optimization runs).

These results will be used for sanity check. In case better results are obtained as a result of GMU benchmarking, only these results will be reported. In case worse results are obtained as a result of GMU benchmarking, the authors of the implementations may be contacted with the requests for providing the applied options of tools.

The FPGA results should be reported for the specific FPGA devices, from two major vendors, Xilinx and Altera, listed below.

High-speed implementations:

Vendor	Family	Device Code
Xilinx	Virtex-6	xc6vlx240tff1156-3 (xc6vlx240t-3ff1156)
	Virtex-7	xc7vx485tffg1761-3 (xc7vx485t-3ffg1761)
Altera	Stratix IV	ep4se530h35c2
	Stratix V	5sgxea7k2f40c1

Lightweight implementations:

Vendor	Family	Device Code
Xilinx	Spartan-6	xc6slx16csg324-3 (xc6slx16-3csg324)
	Artix-7	xc7a100tcsg324-3 (xc7a100t-3csg324)
Altera	Cyclone IV E	ep4ce22f17c6
	Cyclone V E	5ceba4f23c7

These devices have been selected based on their use in popular prototyping boards from Xillinx, Altera, Digilent, and Terasic. In each case, the speed grade has been increased to the maximum possible value, in order to make the results optimal and representative for the entire FPGA family.

For each FPGA device please report *at least* the maximum clock frequency and the resource utilization, including the numbers of

- LUTs, FFs, Slices, BRAMs (should be 0), and DSP slices (should be 0) for Xilinx FPGAs, and
- LEs or ALUTs, FFs, ALMs, embedded memory in Kb (should be 0), and 18x18 multipliers (should be 0) for Altera FPGAs.

All results should be placed in a single file in the Excel, PDF, or ASCII format.

For your reference, we list below major resources available in each of these devices:

Xilinx FPGAs:

Family	Virtex-6	Virtex-7	Spartan-6	Artix-7
Device	xc6vlx240tff1156	xc7vx485tffg1761	xc6slx16csg324	xc7a100tcsg324
LUTs	150,720	303,600	9,112	63,400
Slices	37,680	75,900	2,278	15,850
18Kb BRAMs	832	2,060	32	270
DSP 48E1 Slices	768	2800	32	240
User I/Os	720	700	232	300

Altera FPGAs:

Family	Stratix IV	Stratix V	Cyclone IV E	Cyclone V E
Device	ep4se530h35c2	5sgxea7k2f40c1	ep4ce22f17c6	5ceba4f23c7
LEs	531K	622K	22K	49K
ALMs	212,480	234,720	N/A	18,480
Memory	1,280 M9K 64 M144K	2,560 M20K	594 Kb	308 M10K 485 MLAB
18 x 18 MULs	1,024	512	66	132
User I/Os	744	696	153	224

Other teams are encouraged to perform independent benchmarking for

- The same set of FPGA devices
- A different, independently selected set of FPGA devices
- ASIC technology with various standard-cell libraries.

Submission

Any other materials related to the submitted implementation, e.g., related papers or technical reports, should be placed in the `docs` folder.

The top-level folder should be compressed to a single file

<Authenticated_Cipher_Name>_<Implementation_Team_Name>.zip

e.g., ASCON_GMU.zip.

Either the file itself or its location should be then submitted to the CAESAR's mailing list.