Comparative ASIC Benchmarking of a Group of NIST LWC Candidates

Mustafa Khairallah, Thomas Peyrin and Anupam Chattopadhyay

NTU, Singapore

November 11, 2020

NIST LWC Call for Submissions

Submitted AEAD algorithms and optional hash function algorithms should perform significantly better in constrained environments (**hardware** and embedded software platforms) compared to current NIST standards. They should be optimized to be efficient for short messages (e.g., as short as 8 bytes). **Compact hardware** implementations and embedded software implementations with low RAM and ROM usage should be possible. The performance on **ASIC** and FPGA should consider **a wide range of standard cell libraries**. The algorithms should be flexible to support various implementation strategies (low energy, low power, low latency). The performance on microcontrollers should consider a wide range of 8-bit, 16-bit and 32-bit microcontroller architectures. For algorithms that have a key, the preprocessing of a key (in terms of computation time and memory footprint) should be efficient.

Benchmarking goal

- Compare the baseline performance of different designs?
- Optimize different designs?
- Rank different designs?
- Compare the optimized performance of different designs?

Benchmarking goal

- Compare the baseline performance of different designs?
- Optimize different designs?
- Rank different designs?
- Compare the optimized performance of different designs?

All Valid, but resources are finite.

ASIC Flow

- Front-End: logic and gate level analysis: less accurate, fast, can include more designs.
- Back-End: transistor level analysis, more accurate, slow and includes more details, suitable for second degree analysis.
- Tape-Out: silicon level analysis, real life.

Challenges

- Many designs: 10 designs, 36 implementations, 576 design points.
- Late submissions: the majority of designs submitted very late, partly due to the pandemic.
- Consequently, limited time.
- Different designers = different level of optimizations, different implementation quality, . . .

Approach

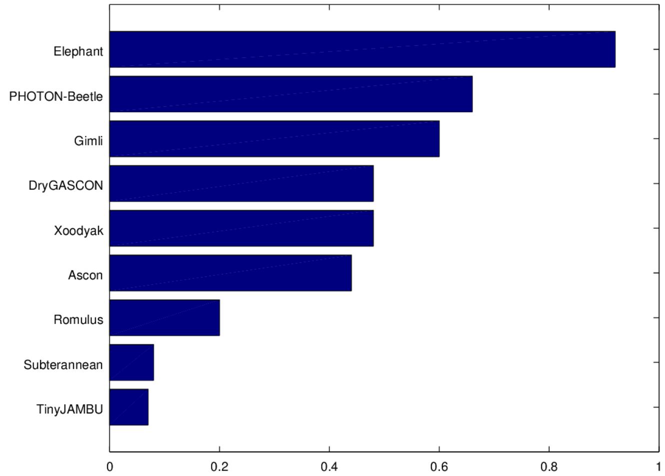
- Design Space Exploration, Round 2: reduce accuracy (front-end), include many designs, and many implementations, study different trade-offs.
- Optimization, Round 3: improve accuracy (back-end), optimize designs (work with the designers on finding better implementations), protected implementations (work with evaluation teams to implement secure implementations).

Ingredients

- Designs: Ascon (1), DryGASCON (1), Elephant (2), Gimli (7), PHOTON-Beetle (1), Pyjamask (2), Romulus (5), Subterranean (2), TinyJAMBU (6) and Xoodyak (8).
- Standard Cell Libraries: TSMC 65nm, FDSOI 28nm.
- Compiler: Synopsys Design Compiler Q-2019.

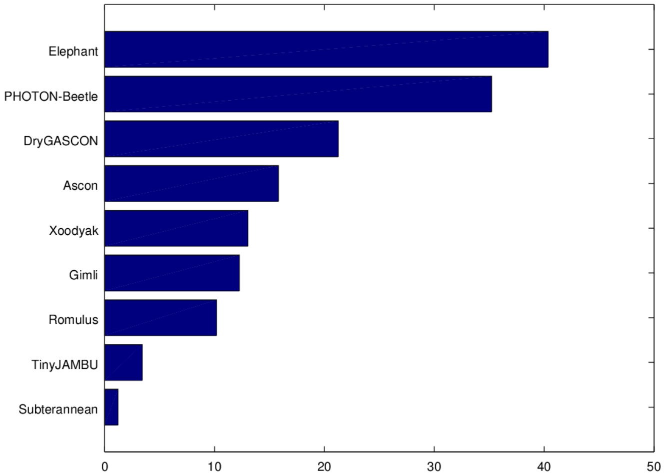
Best Results: TSMC 65nm with CCS circuit models

Energy×Area (Latency×Power×Area) in fJ.KGE, $|A| = |M| = 16$ Bytes



Best Results: TSMC 65nm with CCS circuit models

Energy$\times$Area (Latency$\times$Power$\times$Area) in fJ.KGE, $|A| = |M| = 1536$ Bytes



Conclusions

- Subterranean ranks high on all metrics.
- Romulus and TinyJAMBU are more biased towards low area, energy consumption and efficiency for short messages.
- Ascon, Gimli, and Xoodyak (alphabetically) come next for these metrics, but rank higher for speed over long messages compared to their energy rankings.
- Next is DryGASCON, followed by PHOTON-Beetle and Elephant in most categories.
- Pyjamask (with current implementations) is not suitable for constrained unprotected implementations.