C vs. VHDL: Benchmarking CAESAR Candidates Using High-Level Synthesis and Register-Transfer Level Methodologies

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

http://cryptography.gmu.edu
https://cryptography.gmu.edu/athena
Primary High-Level Synthesis (HLS) Support for This Project

Ekawat Homsirikamol a.k.a “Ice”

Working on the PhD Thesis entitled
“A New Approach to the Development of Cryptographic Standards Based on the Use of High-Level Synthesis Tools”
Register-Transfer Level (RTL) Designs provided by

Ahmed Ferozpuri
Will Diehl
Farnoud Farahmand
“Ice” Homsirikamol

PAEQ
PRIMATEs-APE
PRIMATEs- Gibbon
PRIMATEs-HANUMAN

Minalpher POET SCREAM

AES-COPA CLOC

AES-GCM, ASCON, Deoxys, ICEPOLE, Joltik, Keyak, OCB
Cryptographic Standard Contests

IX.1997

AES

C.2000

NESSIE

XI.2000

CRYPTREC

XII.2002

15 block ciphers $\rightarrow$ 1 winner

I.2000

4 HW winners

+ 4 SW winners

XI.2004

eSTREAM

IV.2008

34 stream ciphers $\rightarrow$

51 hash functions $\rightarrow$ 1 winner

57 authenticated ciphers $\rightarrow$ multiple winners

XI.2004

SHA-3

X.2007

I.2013

CAESAR

XII.2017

time
Evaluation Criteria

- Security
- Software Efficiency
  - µProcessors
  - µControllers
- Hardware Efficiency
  - FPGAs
  - ASICs
- Flexibility
- Simplicity
- Licensing
Traditional Development & Benchmarking Flow


Manual Optimization
- FPGA Tools
- Netlist

Functional Verification
- Timing Verification

Post Place & Route Results

Test Vectors
Extended Traditional Development & Benchmarking Flow

1. Informal Specification
   - Manual Design
     - HDL Code
       - Automated Optimization
         - FPGA Tools
           - Netlist
             - Post Place & Route Results
   - Test Vectors
     - Functional Verification
       - Xilinx ISE + ATHENA
         - Vivado + Default Strategies
     - Timing Verification
Remaining Difficulties of Hardware Benchmarking

- Large number of candidates
- Long time necessary to develop and verify RTL (Register-Transfer Level) Hardware Description Language (HDL) codes
- Multiple variants of algorithms (e.g., multiple key, nonce, and tag sizes)
- High-speed vs. lightweight algorithms
- Multiple hardware architectures
- Dependence on skills of designers
High-Level Synthesis (HLS)

High Level Language
(e.g. C, C++, SystemC)

High-Level Synthesis

Hardware Description Language
(e.g., VHDL or Verilog)
AutoESL Design Technologies, Inc. (25 employees)

Flagship product:

AutoPilot, translating C/C++/System C to VHDL or Verilog

- Acquired by the biggest FPGA company, Xilinx Inc., in 2011
- AutoPilot integrated into the primary Xilinx toolset, Vivado, as Vivado HLS, released in 2012

“High-Level Synthesis for the Masses”
Our Hypotheses

- Ranking of candidate algorithms in cryptographic contests in terms of their performance in modern FPGAs & All-Programmable SoCs will remain the same independently whether the HDL implementations are developed manually or generated automatically using High-Level Synthesis tools.

- The development time will be reduced by at least an order of magnitude.
Potential Additional Benefits

Early feedback for designers of cryptographic algorithms

- Typical design process based only on security analysis and software benchmarking
- Lack of immediate feedback on hardware performance
- Common unpleasant surprises, e.g.,
  - Mars in the AES Contest
  - BMW, ECHO, and SIMD in the SHA-3 Contest
Proposed HLS-Based Development and Benchmarking Flow

Reference Implementation in C

Manual Modifications (pragmas, tweaks)

HLS-ready C code

High-Level Synthesis

HDL Code

Automated Optimization

FPGA Tools

Netlist

Functional Verification

Timing Verification

Xilinx ISE + ATHENa

Vivado + Default Strategies

Test Vectors

Post Place & Route Results
Examples of Source Code Modifications

Unrolling of loops:

```c
for (i = 0; i < 4; i++)
#pragma HLS UNROLL
    for (j = 0; j < 4; j++)
#pragma HLS UNROLL
    b[i][j] = s[i][j];
```

Function Reuse:

```c
// (a) Before modification
for (round=0; round<NB_ROUNDS; ++round)
{
    if (round == NB_ROUNDS-1)
        single_round(state, 1);
    else
        single_round(state, 0);
}
```

Flattening function's hierarchy:

```c
void KeyUpdate (word8 k[4][4],
               word8 round)
{
    #pragma HLS INLINE
    ...
}
```

```c
// (b) After modification
for (round=0; round<NB_ROUNDS; ++round)
{
    if (round == NB_ROUNDS-1)
        x = 1;
    else
        x = 0;
    single_round(state, x);
}
```
Our Test Case

- 13 Round 1 CAESAR candidates + current standard AES-GCM (2 more in progress)
- Basic iterative architecture
- GMU AEAD Hardware API
- Key scheduling and padding done in hardware
- Implementations developed in parallel using RTL and HLS methodology
- Starting point: Informal specifications and reference software implementations in C provided by the algorithm authors
- Post P&R results generated for
  - Xilinx Virtex 6 using Xilinx ISE + ATHENa, and
  - Virtex 7 and Zynq 7000 using Xilinx Vivado with 25 default option optimization strategies
- No use of BRAMs or DSP Units in AEAD Core
## Parameters of Authenticated Ciphers

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Key size</th>
<th>Nonce size</th>
<th>Tag size</th>
<th>Basic Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Block Cipher Based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AES-COPA</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>AES</td>
</tr>
<tr>
<td>AES-GCM</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>AES</td>
</tr>
<tr>
<td>CLOC</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>AES</td>
</tr>
<tr>
<td>Deoxys≠</td>
<td>128</td>
<td>64</td>
<td>128</td>
<td>Deoxys-BC (AES)</td>
</tr>
<tr>
<td>Joltik</td>
<td>128</td>
<td>32</td>
<td>64</td>
<td>Joltik-BC</td>
</tr>
<tr>
<td>OCB</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>AES</td>
</tr>
<tr>
<td>POET</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>AES</td>
</tr>
<tr>
<td>SCREAM</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>TLS</td>
</tr>
</tbody>
</table>
## Parameters of Authenticated Ciphers

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Key size</th>
<th>Nonce size</th>
<th>Tag size</th>
<th>Basic Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCON</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>SPN</td>
</tr>
<tr>
<td>ICEPOLE</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>Keccak-like</td>
</tr>
<tr>
<td>Keyak</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>Keccak-f</td>
</tr>
<tr>
<td>PAEQ</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>AESQ</td>
</tr>
<tr>
<td>PRIMATEs-GBBON</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>PRIMATE</td>
</tr>
<tr>
<td>PRIMATEs-HANUMAN</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>PRIMATE</td>
</tr>
</tbody>
</table>
AEAD Interface

- **PDI** (Public Data Input Ports)
- **SDI** (Secret Data Input Ports)
- **DO** (Data Output Ports)

### Ports
- **clk**, **rst**
- **pdi**, **pdi_valid**, **pdi_ready**
- **sdı**, **sdı_valid**, **sdı_ready**
- **do**, **do_valid**, **do_ready**
### Parameters of Ciphers & GMU Implementations

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Word Size, $w$</th>
<th>Block Size, $b$</th>
<th>#Rounds</th>
<th>Cycles/Block RTL</th>
<th>Cycles/Block HLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block-cipher Based</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AES-COPA</td>
<td>32</td>
<td>128</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>AES-GCM</td>
<td>32</td>
<td>128</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>CLOC</td>
<td>32</td>
<td>128</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>Deoxys</td>
<td>32</td>
<td>128</td>
<td>14</td>
<td>29</td>
<td>32</td>
</tr>
<tr>
<td>Joltik</td>
<td>32</td>
<td>128</td>
<td>32</td>
<td>65</td>
<td>70</td>
</tr>
<tr>
<td>OCB</td>
<td>32</td>
<td>128</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>POET</td>
<td>32</td>
<td>128</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>SCREAM</td>
<td>32</td>
<td>128</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>
## Parameters of Ciphers & GMU Implementations

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Word Size, w</th>
<th>Block Size, b</th>
<th>#Rounds</th>
<th>Cycles/Block RTL</th>
<th>Cycles/Block HLS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Permutation Based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCON</td>
<td>32</td>
<td>64</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>ICEPOLE</td>
<td>256</td>
<td>1024</td>
<td>6</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Keyak</td>
<td>128</td>
<td>1344</td>
<td>12</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>PAEQ</td>
<td>32</td>
<td>368 (M)/240 (AD)</td>
<td>20</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>PRIMATEs-GIBBON</td>
<td>40</td>
<td>40</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>PRIMATEs-HANUMAN</td>
<td>40</td>
<td>40</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>
Datapath vs. Control Unit

Determines
• Area
• Clock Frequency

Determines
• Number of clock cycles
Encountered Problems

Control Unit suboptimal

• Difficulty in inferring an overlap between completing the last round and reading the next input block
• One additional clock cycle used for initialization of the state at the beginning of each round
• The formulas for throughput:

  HLS:  Throughput = Block_size / ((#Rounds+2) * T_{CLK})

  RTL:  Throughput = Block_size / (#Rounds+C * T_{CLK})

  C=0, 1 depending on the algorithm
RTL vs. HLS Clock Frequency in Virtex 7

The graph compares the clock frequencies between RTL and HLS for various designs in Virtex 7. Each design is represented by a different marker and color, with the X-axis representing RTL frequencies and the Y-axis representing HLS frequencies. The markers include:
- Joltik (1,1)
- ASCON (2,2)
- PRIMATEs-GIBBON (3,3)
- PRIMATEs-HANUMAN (4,4)
- PAEQ (5,6)
- AES-GCM (6,5)
- ICEPOLE-128a (7,10)
- CLOC (8,9)
- Deoxys (9,8)
- OCB (10,11)
- Keyak (11,7)
- POET (12,12)
- AES-COPA (13,13)
- SCREAM (14,14)
RTL vs. HLS Throughput in Virtex 7

- ICEPOLE-128a (1,1)
- Keyak (2,2)
- PAEQ (3,3)
- AES-GCM (4,4)
- ASCON (5,5)
- CLOC (6,6)
- OCB (7,7)
- POET (8,8)
- AES-COPA (9,9)
- PRIMATES-GIBBON (10,10)
- SCREAM (11,11)
- Deoxys (12,12)
- PRIMATES-HANUMAN (13,13)
- Joltik (14,14)
RTL vs. HLS Ratios in Virtex 7

Clock Frequency

Throughput
RTL vs. HLS #LUTs in Virtex 7

![Graph showing LUT usage comparison between RTL and HLS for various benchmarks.](image)
RTL vs. HLS Throughput/#LUTs in Virtex 7
RTL vs. HLS Ratios in Virtex 7

#LUTs

Throughput/#LUTs
Throughput vs. LUTs in Virtex 7

**RTL**

**HLS**
RTL vs. HLS #LUTs

![Graph showing LUT usage comparison between RTL and HLS for Virtex 6, Virtex 7, and Zynq]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Virtex 6</th>
<th>Virtex 7</th>
<th>Zynq</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCON</td>
<td>(1,1)</td>
<td>(2,2)</td>
<td>(2,2)</td>
</tr>
<tr>
<td>Joltik</td>
<td>(2,2)</td>
<td>(1,1)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>PRIMATEs HANUMAN</td>
<td>(3,3)</td>
<td>(3,4)</td>
<td>(4,3)</td>
</tr>
<tr>
<td>PRIMATEs GIBBON</td>
<td>(4,4)</td>
<td>(4,3)</td>
<td>(3,4)</td>
</tr>
<tr>
<td>Deoxys</td>
<td>(5,7)</td>
<td>(5,5)</td>
<td>(5,5)</td>
</tr>
<tr>
<td>AES-GCM</td>
<td>(6,6)</td>
<td>(6,6)</td>
<td>(6,6)</td>
</tr>
<tr>
<td>SCREAM</td>
<td>(7,8)</td>
<td>(7,9)</td>
<td>(9,9)</td>
</tr>
<tr>
<td>CLOC</td>
<td>(8,5)</td>
<td>(8,7)</td>
<td>(7,7)</td>
</tr>
<tr>
<td>OCB</td>
<td>(9,9)</td>
<td>(9,8)</td>
<td>(8,8)</td>
</tr>
<tr>
<td>CEPOLE-128a</td>
<td>(10,10)</td>
<td>(10,10)</td>
<td>(10,10)</td>
</tr>
<tr>
<td>Keyak</td>
<td>(11,11)</td>
<td>(11,11)</td>
<td>(11,11)</td>
</tr>
<tr>
<td>PAEQ</td>
<td>(12,12)</td>
<td>(13,12)</td>
<td>(13,12)</td>
</tr>
<tr>
<td>POET</td>
<td>(13,13)</td>
<td>(12,13)</td>
<td>(12,13)</td>
</tr>
<tr>
<td>AES-COPA</td>
<td>(14,14)</td>
<td>(14,14)</td>
<td>(14,14)</td>
</tr>
</tbody>
</table>
RTL vs. HLS Throughput/#LUTs
ATHENa Database of Results for Authenticated Ciphers

- Available at
  http://cryptography.gmu.edu/athena

- Developed by John Pham, a Master’s-level student of Jens-Peter Kaps

- Results can be entered by designers themselves. If you would like to do that, please contact us regarding an account.

- The ATHENa Option Optimization Tool supports automatic generation of results suitable for uploading to the database
### 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
**Ranking View (2)**

Throughput for:  
- Authenticated Encryption
- Authenticated Decryption
- Authentication Only

Min Area: 0  
Max Area: 1000000
Min Throughput: 0  
Max Throughput: 100000
Source:  
- Source Available

Ranking:  
- Throughput/Area
- Throughput
- Area

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

Show 25 entries

<table>
<thead>
<tr>
<th>Result ID</th>
<th>Algorithm</th>
<th>Key Size [bits]</th>
<th>Implementation Approach</th>
<th>Hardware API</th>
<th>Arch Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>154</td>
<td>ICEPOLE</td>
<td>128</td>
<td>RTL</td>
<td>GMU_AEAD_Core_API_v1.1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>73</td>
<td>Keyek</td>
<td>128</td>
<td>RTL</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>62</td>
<td>AES-GCM</td>
<td>128</td>
<td>RTL</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>65</td>
<td>CLOC</td>
<td>128</td>
<td>HLS</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>80</td>
<td>PRIMATEs-GIBBON</td>
<td>120</td>
<td>RTL</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>144</td>
<td>OCB</td>
<td>128</td>
<td>RTL</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>124</td>
<td>PRIMATEs-HANUMAN</td>
<td>120</td>
<td>HLS</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>86</td>
<td>SCREAM</td>
<td>128</td>
<td>RTL</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>142</td>
<td>Joltik</td>
<td>128</td>
<td>RTL</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>75</td>
<td>POET</td>
<td>128</td>
<td>RTL</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
<tr>
<td>60</td>
<td>AES-COPA</td>
<td>128</td>
<td>RTL</td>
<td>GMU_AEAD_Core_API_v1</td>
<td>Basic Iterative</td>
</tr>
</tbody>
</table>
Details of Result ID 97

Algorithm
- IV or Nonce Size [bits]: 96
- Transformation Category: Cryptographic
- Transformation: Authenticated Cipher
- Group: Standards
- Algorithm: AES-GCM
- Tag Size [bits]: 128
- Associated Data Support: -
- Key Size [bits]: 128
- Secret Message Number: -
- Secret Message Number Size [bits]: -
- Message Block Size [bits]: 128
- Other Parameters: -
- Specification: SP-800-38D.pdf
- Formula for Message Size After Padding: -

Design
- Design ID: 21
- Impl Approach: HLS
- Hardware API: GMU_AEAD_Core_API_v1
- Primary Optimization Target: Throughput/Area
- Secondary Optimization Target: -
- Architecture Type: Basic Iterative
- Description Language: VHDL
- Use of Megafunctions or Primitives: No
- List of Megafunctions or Primitives: -
- Maximum Number of Streams Processed in Parallel: 1
- Number of Clock Cycles per Message Block in a Long Message: 12
- Datapath Width [bits]: 128
- Padding: Yes
- Minimum Message Unit: -
- Input Bus Width [bits]: 32
- Output Bus Width [bits]: 32
## Comparison of Result #s 95 and 97

### Algorithm
- **IV or Nonce Size [bits]:** 96
- **Transformation Category:** Cryptographic
- **Transformation:** Authenticated Cipher
- **Group:** Standards
- **Algorithm:** AES-GCM
- **Tag Size [bits]:** 128
- **Associated Data Support:** No
- **Key Size [bits]:** 128
- **Secret Message Number:** No
- **Secret Message Number Size [bits]:** No
- **Message Block Size [bits]:** 128
- **Other Parameters:** No
- **Specification:** SP-800-38D.pdf

### Design
- **Design ID:** 20
- **Impl Approach:** RTL
- **Hardware API:** GMU_AEAD_Core_API_v1
- **Primary Optimization Target:** Throughput/Area
- **Secondary Optimization Target:** Throughput/Area
- **Architecture Type:** Basic Iterative
- **Use of Megablocks or Primitives:** No
- **List of Megablocks or Primitives:** No
- **Maximum Number of Streams Processed in Parallel:** 1
- **Number of Clock Cycles per Message Block in a Long Message:** 11
- **Datapath Width [bits]:** 128
- **Padding:** Yes
- **Minimum Message Unit:** 32
- **Input Bus Width [bits]:** 32

<table>
<thead>
<tr>
<th><strong>Platform</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Vendor:</td>
<td>Xilinx</td>
<td>Xilinx</td>
</tr>
<tr>
<td>Family:</td>
<td>Virtex 7</td>
<td>Virtex 7</td>
</tr>
<tr>
<td>Device:</td>
<td>xc7vx485tffg1761-2</td>
<td>xc7vx485tffg1761-2</td>
</tr>
</tbody>
</table>

| **Timing** | 3261 | 3015 |
| Encryption/Authentication Throughput [Mbits/s]: | 3261 | 3015 |
| Decryption/Authentication Throughput [Mbits/s]: | 3261 | 3015 |
| Authentication-Only Throughput [Mbits/s]: | - | - |
| Synthesis Clock Frequency [MHz]: | - | - |
| Key Scheduling Time [ns]: | - | - |
| Requested Implementation Clock Frequency [MHz]: | - | - |
| Implementation Clock Frequency [MHz]: | 280.27 | 282.65 |
| (Encryption/Authentication Throughput)/LUT [(Mbits/s)/LUT]: | 0.909 | 0.879 |
| (Encryption/Authentication Throughput)/Slice [(Mbits/s)/Slice]: | 2.797 | 2.728 |
| (Decryption/Authentication Throughput)/LUT [(Mbits/s)/LUT]: | 0.909 | 0.879 |
| (Decryption/Authentication Throughput)/Slice [(Mbits/s)/Slice]: | 2.797 | 2.728 |
| (Auth-Only Throughput)/LUT [(Mbits/s)/LUT]: | 0.909 | 0.879 |
| (Auth-Only Throughput)/Slice [(Mbits/s)/Slice]: | 2.797 | 2.728 |

| **Resource Utilization** | 1166 | 1105 |
| CLBs: | 3588 | 3430 |
| LUTs: | - | - |
| Flip Flops: | 0 | 0 |
| DSPs: | 0 | 0 |
| BRAMs: | 0 | 0 |
Conclusions

• High-level synthesis offers a potential to facilitate hardware benchmarking during the design of cryptographic algorithms and at the early stages of cryptographic contests

• Case study based on 13 Round 1 CAESAR candidates & AES-GCM demonstrated correct ranking for majority of candidates using all major performance metrics

• More research & development needed to overcome remaining difficulties
  • Suboptimal control unit of HLS implementations
  • Wide range of RTL to HLS performance metric ratios
  • A few potentially suboptimal HLS or RTL implementations
  • Efficient and reliable generation of HLS-ready C codes
Thank you!

Comments?

Questions?

Suggestions?

ATHENa:  http://cryptography.gmu.edu/athena
CERG:  http://cryptography.gmu.edu