Fair and Efficient Hardware Benchmarking of Candidates in Cryptographic Contests

Kris Gaj
CERG
George Mason University

Partially supported by NSF under grant no. 1314540
Designs & results for this talk contributed by

“Ice” Homsirikamol  Farnoud Farahmand  Ahmed Ferozpuri

Will Diehl  Marcin Rogawski  Panasayya Yalla
Cryptographic Standard Contests

- AES (1997-2000)
  - 15 block ciphers → 1 winner
- NESSIE (2000-2002)
- CRYPREC (2002-2004)
- eSTREAM (2004-2008)
- SHA-3 (2008-2013)
  - 51 hash functions → 1 winner
  - 57 authenticated ciphers → multiple winners

34 stream ciphers → 4 HW winners + 4 SW winners

Time:
- IX.1997
- X.2000
- I.2000
- XII.2002
- XI.2004
- IV.2008
- X.2007
- X.2012
- I.2013
- TBD
Evaluation Criteria in Cryptographic Contests

- Security
- Software Efficiency
  - µProcessors
  - µControllers
- Hardware Efficiency
  - FPGAs
  - ASICs
- Flexibility
- Simplicity
- Licensing
AES Contest 1997-2000
Final Round

**Speed in FPGAs**

**Votes at the AES 3 conference**

**GMU results**

**Hardware results matter!**
Throughput vs. Area Normalized to Results for SHA-256 and Averaged over 11 FPGA Families – 256-bit variants
SHA-3 finalists in high-performance FPGA families

Diagram showing normalized throughput/area for different FPGA families:

- Stratix IV
- Stratix III
- Virtex 6
- Virtex 5

Axes:
- Y-axis: Stratix IV, Stratix III, Virtex 6, Virtex 5
- X-axis: Normalized Throughput/Area

Legend:
- BLAKE
- Groestl
- Skein
- JH
- Keccak

Values:
- 0.25, 0.35, 0.50, 0.79, 1.00, 1.41, 2.00, 2.83, 4.00
- $2^{-2}, 2^{-1.5}, 2^{-1}, 2^{-0.5}, 2^0, 2^{0.5}, 2^1, 2^{1.5}, 2^2$
# FPGA Evaluations – From AES to SHA-3

<table>
<thead>
<tr>
<th></th>
<th>AES</th>
<th>eSTREAM</th>
<th>SHA-3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Primary optimization target</td>
<td>Throughput</td>
<td>Area Throughput/Area</td>
<td>Throughput/Area</td>
</tr>
<tr>
<td>Multiple architectures</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Embedded resources</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Benchmarking</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple FPGA families</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Specialized tools</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Experimental results</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Reproducibility</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability of source codes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Database of results</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Hardware Benchmarking in Cryptographic Contests

- **Focus on ranking**, rather than absolute values
- Only relatively **large differences** (>20-30%) matter
- Winner in use for the next 20-30 years, implemented using **technologies not in existence** today
- Very **wide range of possible applications**, and as a result performance and cost targets
- **Large number** of candidates
- **Limited time** for evaluation
- Results are **final**
### Number of Candidates in Cryptographic Contests

<table>
<thead>
<tr>
<th>Cryptography Standard</th>
<th>Initial number of candidates</th>
<th>Implemented in hardware</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>15</td>
<td>5</td>
<td>33.3%</td>
</tr>
<tr>
<td>eSTREAM</td>
<td>34</td>
<td>8</td>
<td>23.5%</td>
</tr>
<tr>
<td>SHA-3</td>
<td>51</td>
<td>14</td>
<td>27.5%</td>
</tr>
</tbody>
</table>
CAESAR Competition

Goal: Portfolio of new-generation authenticated ciphers

First-round submissions: March 15, 2014

Announcement of final portfolio: TBD

Organizer: An informal committee of leading cryptographic experts

Number of candidate families:

Round 1: 57  Round 2: 29  Round 3: 15
Two Possible Approaches

Better Manual Register Transfer Level

Automated High-Level Synthesis
Manual Register-Transfer Level Approach
New in CAESAR Hardware Benchmarking

CAESAR Committee:

1) Design teams asked to submit their own Verilog/VHDL code
2) Three Use Cases corresponding to different applications and different optimization targets

GMU Benchmarking Team:

1) Standard hardware Application Programming Interface (API)
2) Comprehensive Implementer’s Guide and Development Packages, including VHDL and Python code common for all candidates
CAESAR Hardware API

Minimum Compliance Criteria

- Encryption, decryption, key scheduling
- Padding
- Maximum size of message & AD
- Permitted data port widths, etc.

Communication Protocol

Interface

Timing Characteristics
CAESAR Hardware API vs. GMU Development Package

CAESAR Hardware API:

1) Approved by the CAESAR Committee in May 2016, **stable**
2) Necessary for **fairness** and **compatibility**
3) **Obligatory**

GMU Development Package:

1) First version published in May 2016, constantly **evolving**
2) Recommended in order to reduce the **development time**
3) **Totally optional**
Top-level block diagram of a High-Speed architecture
Development Package

a. VHDL code of a generic PreProcessor, PostProcessor, and CMD FIFO, common for all CAESAR Candidates (src_rtl)
b. Universal testbench common for all CAESAR candidates (AEAD_TB)
c. Python app used to automatically generate test vectors (aeadtvgen)
d. Reference implementations of Dummy authenticated ciphers (dummyN)

To be extended with support for lightweight implementations in June 2017
The API Compliant Code Development

Development Package `dummyN`

Development Package `src_rtl`

Specification

Reference C Code

Development Package `aeadtvgen`

Test Vectors

Formulas for the Execution Time & Throughput

Manual Design

HDL Code

FPGA Tools

Post Place & Route Results (Resource Utilization, Max. Clock Frequency)

Functional Verification

Development Package `AEAD_TB`

Pass/ Fail
Round 2 VHDL/Verilog Submitters

1. CCRG NTU (Nanyang Technological University) Singapore – ACORN, AEGIS, JAMBU, & MORUS
2. CLOC-SILC Team, Japan – CLOC & SILC
3. Ketje-Keyak Team – Ketje & Keyak
4. Lab Hubert Curien, St. Etienne, France – ELmD & TriviA-ck
5. Axel Y. Poschmann and Marc Stöttinger – Deoxys & Joltik
6. NEC Japan – AES-OTR
7. IAIK TU Graz, Austria – Ascon
8. DS Radboud University Nijmegen, Netherlands – HS1-SIV
9. IIS ETH Zurich, Switzerland – NORX
10. Pi-Cipher Team – Pi-Cipher
11. EmSec RUB, Germany – POET
12. CG UCL, INRIA – SCREAM
13. Shanghai Jiao Tong University, China – SHELL

Total: 19 Candidate Families
Round 2 VHDL Submitters – GMU Team

“Ice” Homsirikamol
AES-GCM, AEZ, Ascon, Deoxys, HS1-SIV, ICEPOLE, Joltik, NORX, OCB, PAEQ, Pi-Cipher, STRIBOB

Will Diehl
Minalpher
OMD
POET
SCREAM

Ahmed Ferozpurri
PRIMATEs-GIBBON & HANUMAN, PAEQ

Farnoud Farahmand
AES-COPA CLOC

Mike X. Lyons
TriviA-ck

Total: 19 Candidate Families + AES-GCM
CAESAR Round 2 Statistics

• 75 unique designs
• Covering the majority of primary variants of
  28 out of 29 Round 2 Candidate Families (all except Tiaoxin)
• High-speed implementation of AES-GCM (baseline)

The biggest and the earliest hardware benchmarking effort in the history of cryptographic competitions
RTL Benchmarking

Optimal Options of Tools (for the best Throughput/Area)

 Automated Optimization

 Post Place & Route Results (Resource Utilization, Max. Clock Frequency)

 HDL Code

 FPGA Tools

 Replication Script

 4 high-performance &
 4 low-cost
 FPGA Families
 from Xilinx & Altera
Generation of Results Facilitated by ATHENa

“working” with ATHENa...

old days...
ATHENa – Automated Tool for Hardware Evaluation

- Open-source
- Written in Perl
- Developed 2009-2012
- FPL Community Award 2010
- Automated search for optimal
  - Options of tools
  - Target frequency
  - Starting placement point
- Supporting Xilinx ISE, Altera Quartus

No support for Xilinx Vivado
Extension of ATHENa to Vivado: Minerva

- **Programming language:** Python
- **Target synthesis and implementation tool:** Xilinx Vivado Design Suite
- **Supported FPGA families:** All Xilinx 7 series and beyond
- **Optimization criteria:**
  1. Maximum frequency
  2. Frequency/#LUTs
  3. Frequency/#Slices

Expected release – July 2017
Relative Improvement of Results (Throughput/Area) by Using Minerva: Virtex 7, Round 2 CAESAR Candidates

Ratios of results obtained using Minerva vs. binary search
Relative Throughput/Area in Virtex 6 vs. AES-GCM

E – Throughput/Area for Encryption
D – Throughput/Area for Decryption
A – Throughput/Area for Authentication Only
Default: Throughput/Area the same for all 3 operations

Red – algorithms qualified to Round 3

Throughput/Area of AES-GCM = 1.020 (Mbit/s)/LUTs
Relative Throughput in Virtex 6
Ratio of a given Cipher Throughput/Throughput of AES-GCM

Throughput of AES-GCM = 3239 Mbit/s

E – Throughput for Encryption
D – Throughput for Decryption
A – Throughput for Authentication Only
Default: Throughput the same for all 3 operations
ATHENa Database of Results

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

- Developed by John Pham, a Master’s-level student of Jens-Peter Kaps as a part of the SHA-3 Hardware Benchmarking project, 2010-2012, (sponsored by NIST)

- In June 2015 extended to support Authenticated Ciphers

  Extension to support Round 3 Use Cases and ranking of candidate variants - Summer 2017
## Number of Candidates in Cryptographic Contests

<table>
<thead>
<tr>
<th></th>
<th>Initial number of candidates</th>
<th>Implemented in hardware</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AES</strong></td>
<td>15</td>
<td>5</td>
<td>33.3%</td>
</tr>
<tr>
<td><strong>eSTREAM</strong></td>
<td>34</td>
<td>8</td>
<td>23.5%</td>
</tr>
<tr>
<td><strong>SHA-3</strong></td>
<td>51</td>
<td>14</td>
<td>27.5%</td>
</tr>
<tr>
<td><strong>CAESAR</strong></td>
<td>57</td>
<td>28</td>
<td>49.1%</td>
</tr>
</tbody>
</table>
Remaining Problems

- Different skills of designers
- Different amount of time and effort
- Misunderstandings regarding Hardware API
- Requests for extending the deadline or disregarding ALL results
High-Level Synthesis Approach
Potential Solution: High-Level Synthesis (HLS)

- High Level Language (preferably C or C++)
- High-Level Synthesis
- Hardware Description Language (VHDL or Verilog)
Case for High-Level Synthesis & Crypto Contests

- Each submission includes reference implementation in C
- Development time potentially decreased several times
- All candidates can be implemented by the same group, and even the same designer
- Results from High-Level Synthesis could have a large impact in early stages of the competitions and help narrow down the search
- RTL code and results from previous contests form excellent benchmarks for High-Level Synthesis tools, which can generate fast progress targeting cryptographic applications
BEFORE: Early feedback for designers of 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

DURING: Faster design space exploration

- Multiple hardware architectures (folded, unrolled, pipelined, etc.)
- Multiple variants of the same algorithms (e.g., key, nonce, tag size)
- Detecting suboptimal manual designs
Our Hypotheses

• **Ranking** of candidates in cryptographic contests in terms of their performance in modern FPGAs 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 a factor of 3 to 10.

• This hypothesis **should apply to** at least:
  - AES Contest, SHA-3 Contest, CAESAR Contest
  - possibly Post-quantum Cryptography?
In-Use Tools supporting C, C++, Extended C

Commercial:

- **Vivado HLS**: Xilinx
- **CHC**: Altium; **CoDeveloper**: Impulse Accelerated; **Cynthesizer**: FORTE; **eXCite**: Y Explorations; **ROCCC**: Jacquard Comp.
- **Catapult-C**: Calypto Design Systems; **CtoS**: Cadence; **DK Design Suite**: Mentor Graphics; **Symphony C**: Synopsys

Academic:

- **Bambu**: Politecnico di Milano, Italy
- **DWARV**: Delft University of Technology, The Netherlands
- **GAUT**: Universite de Bretagne-Sud, France
- **LegUp**: University of Toronto, Canada
Our Choice of the HLS Tool: Vivado HLS

- **Integrated** into the primary Xilinx toolset, Vivado, and released in 2012
- **Free** (or almost free) licenses for academic institutions
- Good **documentation** and user **support**
- The largest number of **performance optimizations**
- On average the **highest clock frequency** of the generated code
Licensing Limitations of Vivado HLS

1. Designers are not allowed to target devices of other FPGA vendors (e.g., Altera)
2. Designers are not allowed to target ASICs
3. Results cannot be compared with results obtained using other HLS tools
Our Approach: Language Partitioning
Transformation to HLS-ready C/C++ Code

1. Addition of HLS Tool directives (pragmas)
2. Hardware-driven code refactoring
3. Mapping software to hardware API
Code Refactoring – High-Level

Reference C

HLS-ready C/C++

Encryption

Decryption

Encryption/Decryption

Use of pragmas possible but unreliable
Code Refactoring: Low-Level

Single vs. Multiple Function Calls:

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

// (b) After modification
for(round=0; round<NB_ROUNDS; ++round)
{
    if (round == NB_ROUNDS-1)
        x = 1;
    else
        x = 0;
    single_round(state, x);
}
Sources of Productivity Gains

- Higher-level of abstraction
- Focus on datapath rather than control logic
- Debugging in software (C/C++)
  - Faster run time
  - No timing waveforms
GMU Case Studies

• 5 Final SHA_3 Candidates
  Applied Reconfigurable Computing, ARC 2015, Bochum
  [paper + presentation]

• 18 Round 2 CAESAR Candidates + AES-GCM
  Directions in Authenticated Ciphers, DIAC 2016, Nagoya, Japan
  [presentation only]

• 15 Round 3 CAESAR Candidates + AES-GCM
  (all Round 3 families except Keyak and AEZ)
  [this talk]
Ending Point for Optimization

Target: Basic Iterative Architecture

- Initial \#cycles\_per\_block: \textbf{thousands}
- Expected \#cycles\_per\_block: \#rounds + \(\varepsilon\), \(\varepsilon = 0 - 2\)

- Examples of results achieved for Round 3 Candidates:

  \#rounds + 0: ACORN, AEGIS, MORUS, Ketje, Tiaoxin
  (all with \#rounds = 1)

  \#rounds + 2: AES-GCM, AES-OTR, Ascon, COLM,
  Deoxys, JAMBU-AES, NORX, OCB, SILC-AES

No need for the RTL implementation or timing analysis before HLS implementation
RTL vs. HLS Ratios for Throughput/Area in Virtex 6

- **Suboptimal HLS**: > 1.30
- **[0.90, 1.30]**: RTL and HLS acceptable
- **(0.70, 0.90]**: RTL may be improved
- **< 0.70**: Suboptimal RTL

<table>
<thead>
<tr>
<th>Component</th>
<th>RT/HT Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>SILC</td>
<td>1.91</td>
</tr>
<tr>
<td>Ascon</td>
<td>1.66</td>
</tr>
<tr>
<td>NORX</td>
<td>1.60</td>
</tr>
<tr>
<td>OCB</td>
<td>1.51</td>
</tr>
<tr>
<td>Tiaxin</td>
<td>1.34</td>
</tr>
<tr>
<td>JAMBU-AES</td>
<td>1.28</td>
</tr>
<tr>
<td>Deoxys</td>
<td>1.19</td>
</tr>
<tr>
<td>COLM</td>
<td>1.17</td>
</tr>
<tr>
<td>CLOC</td>
<td>1.01</td>
</tr>
<tr>
<td>AES-GCM</td>
<td>1.00</td>
</tr>
<tr>
<td>Ketje</td>
<td>1.00</td>
</tr>
<tr>
<td>JAMBU-SIMON</td>
<td>0.93</td>
</tr>
<tr>
<td>MORUS</td>
<td>0.92</td>
</tr>
<tr>
<td>AEGIS</td>
<td>0.88</td>
</tr>
<tr>
<td>ACORN</td>
<td>0.81</td>
</tr>
<tr>
<td>AES-OTR</td>
<td>0.46</td>
</tr>
</tbody>
</table>
RTL vs. HLS Throughput/Area [((Mbits/s)/LUTs)]

Consistently better than AES-GCM

Suboptimal HLS

Suboptimal RTL
Conclusions

Accuracy:
• **Good** (but not perfect) correlation between algorithm rankings using RTL and HLS approaches

Efficiency:
• Changes in the reference code (**code refactoring**) needed to infer the desired architecture, and thus #cycles_per_block
• **3-10 improvement** in the development time
• Designer can **focus on functionality** : control logic inferred
• Much **easier verification** : C/C++ testbenches
• **A single designer** can produce implementations of multiple (and even all) candidates

Bottom Line:
• Manual (RTL) design approach still predominant
• HLS design approach at the experimental stage – more research needed
Future Work

1. Step-by-step **designer’s guide**, including **general strategies** for
   - Code refactoring
   - Pragmas insertion

2. Multiple **examples** (AES, SHA-3, CAESAR contests)

3. Use for **design space exploration** (folding, unrolling, etc.)

4. Experiments with **multiple HLS tools**
   - automated **translation of Vivado HLS pragmas** to pragmas of leading academic tools: Bambu, DWARV, LegUp

5. Identifying **limitations of HLS tools** and possible improvements
Possible Future Uses of HLS

Identifying *suboptimal RTL implementations* in Round 3 of the CAESAR Contest

Designing *new building blocks* [e.g., rounds, steps, etc.] *for* hardware-friendly block ciphers, hash functions, and authenticated ciphers

Post-Quantum Cryptography

Early Rounds of Future Contests
Thank you!

Questions?

Comments?

Suggestions?

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