C vs. VHDL: Comparing Performance of CAESAR Candidates Using High-Level Synthesis on Xilinx FPGAs

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

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

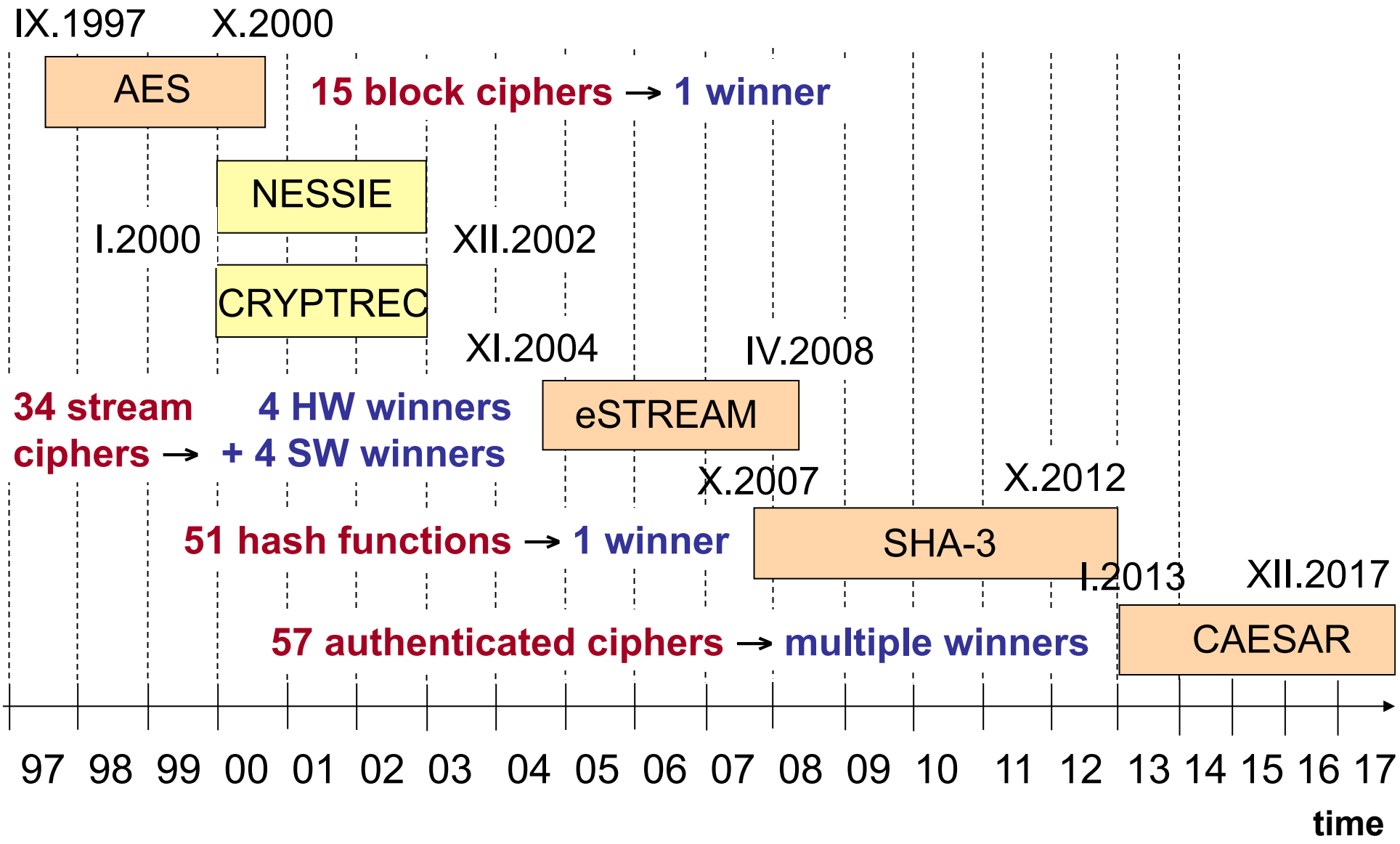
Primary Support for This Particular Project

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"

RTL codes developed by:
William Diehl,
Farnoud Farahmand,
Ahmed Ferozpuri, and
Ekawat Homsirikamol.

Cryptographic Standard Contests



Evaluation Criteria

Security

Software Efficiency

µProcessors µControllers

Hardware Efficiency

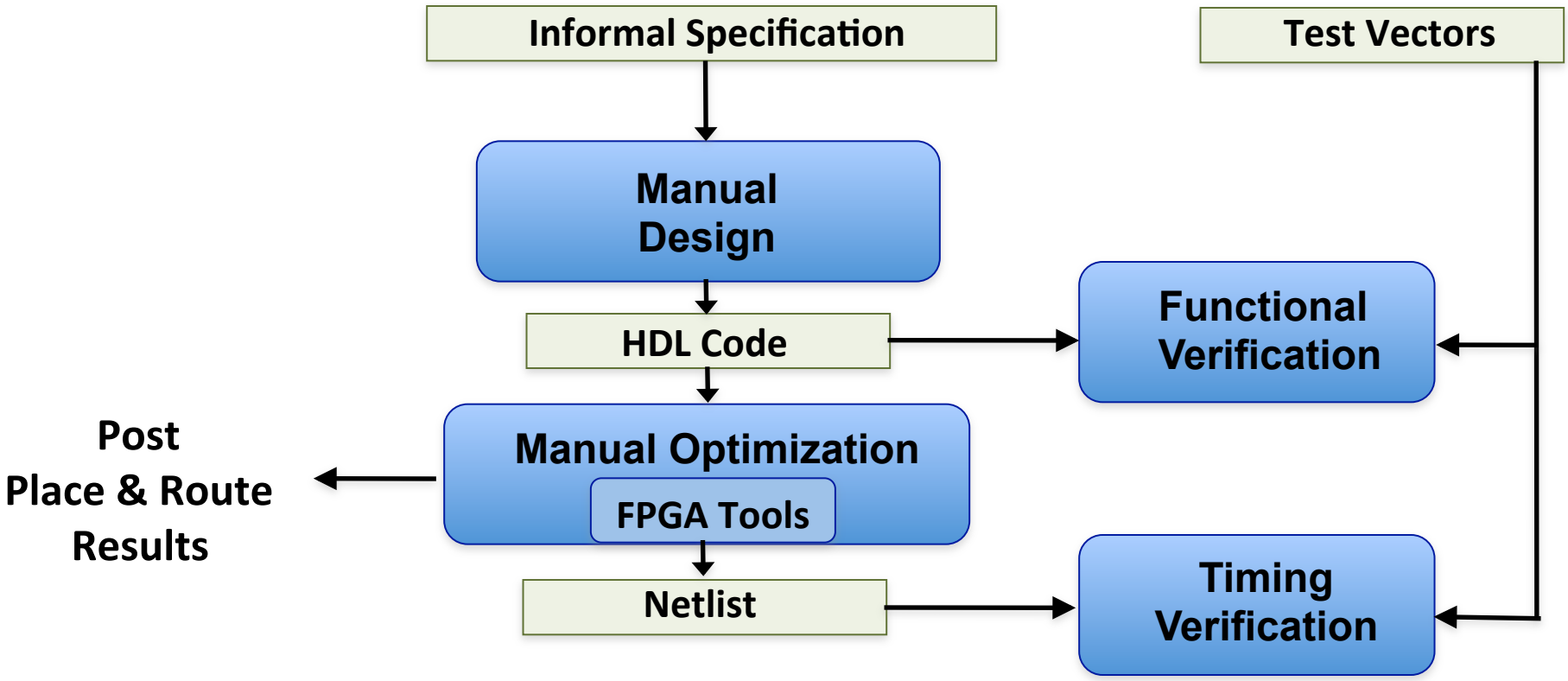
FPGAs ASICs

Flexibility

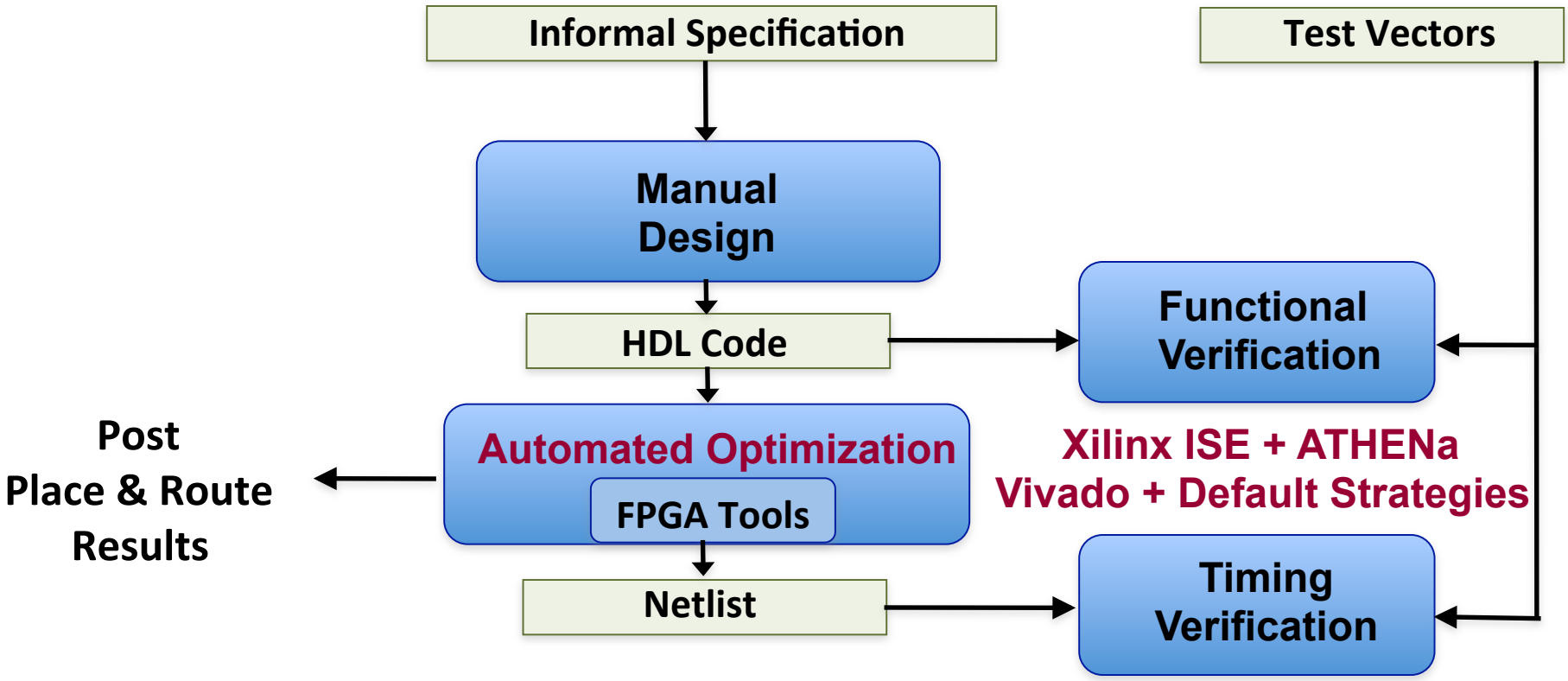
Simplicity

Licensing

Traditional Development & Benchmarking Flow



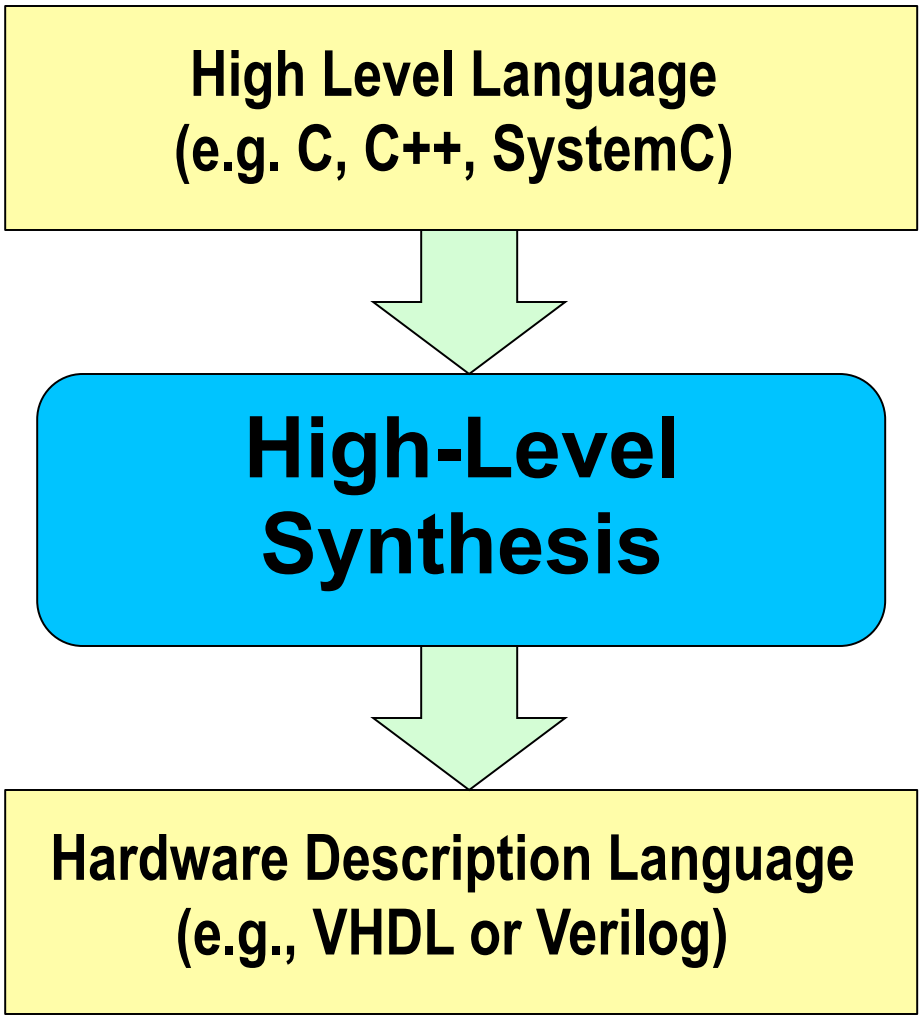
Extended Traditional Development & Benchmarking Flow



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)



Short History of High-Level Synthesis

G. Martin & G. Smith "HLS: Past, Present, and Future," IEEE D&ToC, 2009

Generation 1 (1980s-early 1990s): research period

Generation 2 (mid 1990s-early 2000s):

- **Commercial tools from Synopsys, Cadence, Mentor Graphics, etc.**
- **Input languages: behavioral HDLs Target: ASIC**

Outcome: Commercial failure

Generation 3 (from early 2000s):

- **Domain oriented commercial tools: in particular for DSP**
- **Input languages: C, C++, C-like languages (Impulse C, Handel C, etc.), Matlab + Simulink, Bluespec**
- **Target: FPGA, ASIC, or both**

Outcome: First success stories

Cinderella Story

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**

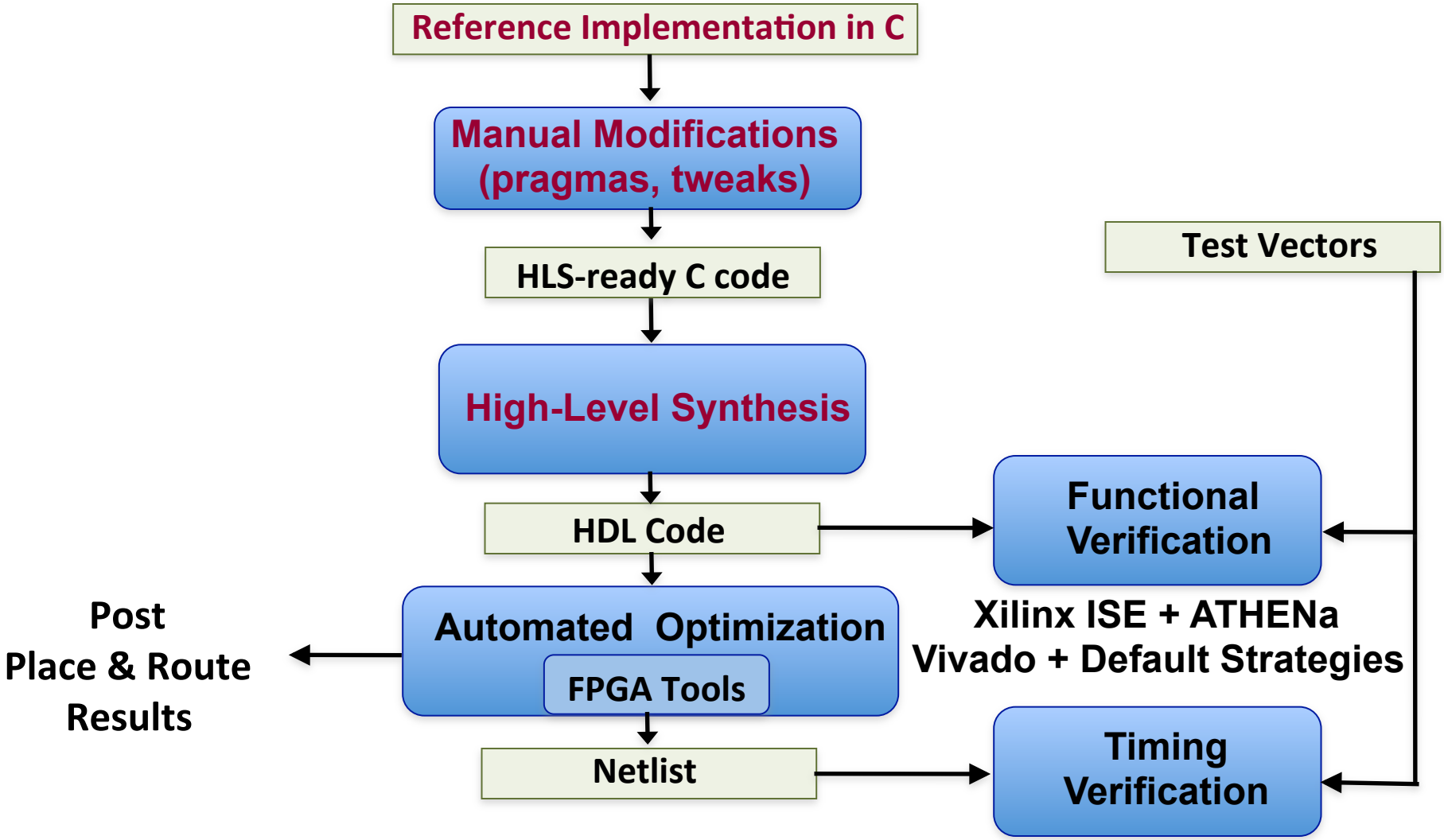
- **The development time will be reduced by at least an order of magnitude**

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



Examples of Source Code Modifications

Unrolling of loops:

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

Flattening function's hierarchy:

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

Function Reuse:

```
// (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);
  }
```

Our Test Case

- 8 Round 1 CAESAR candidates + current standard AES-GCM
- Basic iterative architecture
- GMU AEAD Hardware API
- Implementations developed in parallel using RTL and HLS methodology
- 2-3 RTL implementations per student, all HLS implementations developed by a single student (Ice)
- 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 26 default option optimization strategies
- No use of BRAMs or DSP Units in AEAD Core

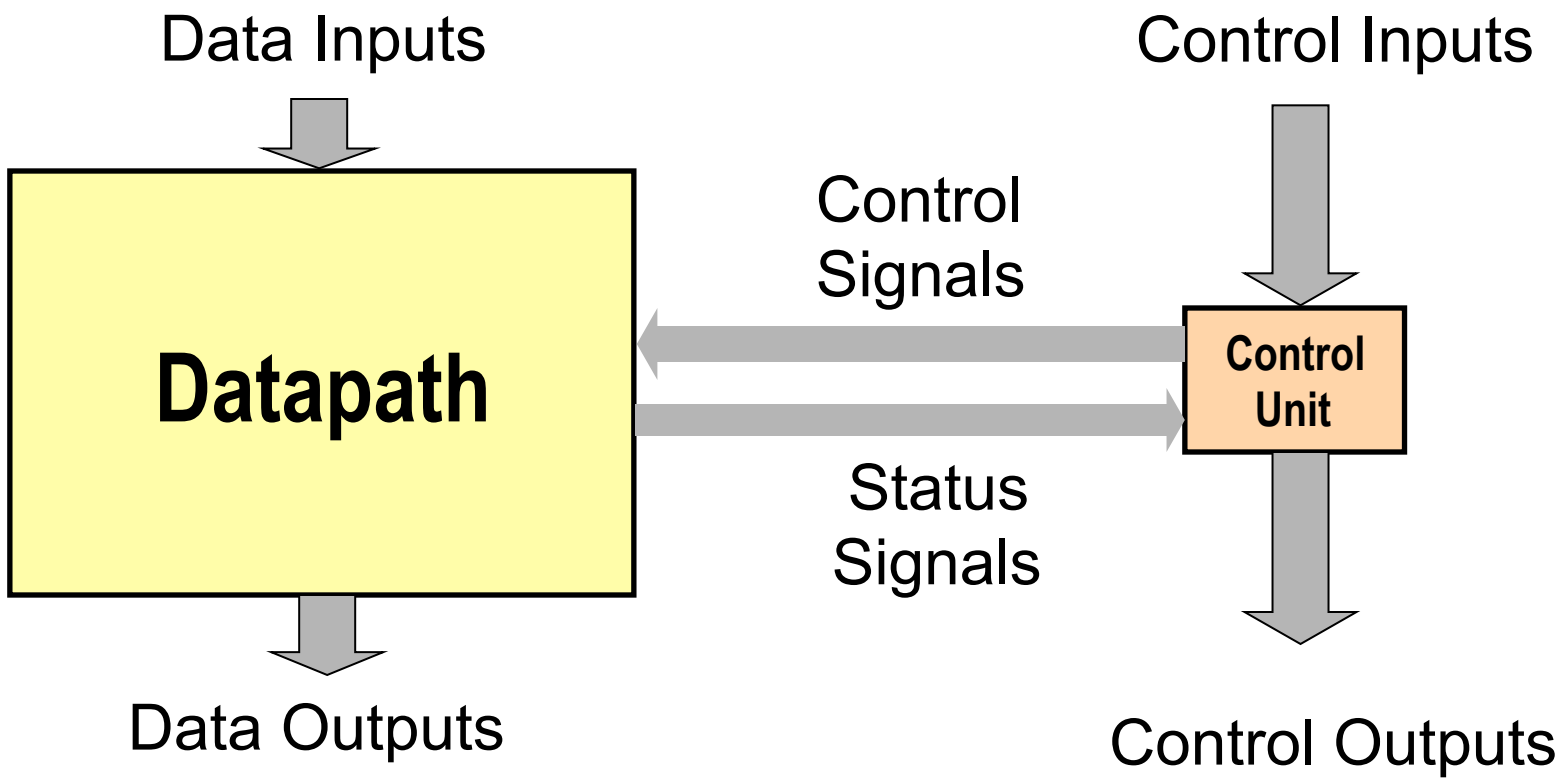
Parameters of Authenticated Ciphers

Algorithm	Key size	Nonce size	Tag size	Basic Primitive
		Block Cipher Based		
AES-COPA	128	128	128	AES
AES-GCM	128	96	128	AES
CLOC	128	96	128	AES
POET	128	128	128	AES
SCREAM	128	96	128	TLS
		Permutation Based		
ICEPOLE	128	128	128	Keccak-like
Keyak	128	128	128	Keccak-f
PRIMATEs-GIBBON	120	120	120	PRIMATE
PRIMATEs-HANUMAN	120	120	120	PRIMATE

Parameters of Ciphers & GMU Implementations

Algorithm	Word Size, w	Block Size, b	#Rounds	Cycles/Block RTL	Cycles/Block HLS
Block-cipher Based					
AES-COPA	32	128	10	11	12
AES-GCM	32	128	10	11	12
CLOC	32	128	10	11	12
POET	32	128	10	11	12
SCREAM	32	128	10	11	12
Permutation Based					
ICEPOLE	256	1024	6	6	8
Keyak	128	1344	12	12	14
PRIMATEs-GIBBON	40	40	6	7	8
PRIMATEs-HANUMAN	40	40	12	13	14

Datapath vs. Control Unit



Determines

- **Area**
- **Clock Frequency**

Determines

- **Number of clock cycles**

Encountered Problems

Control Unit suboptimal

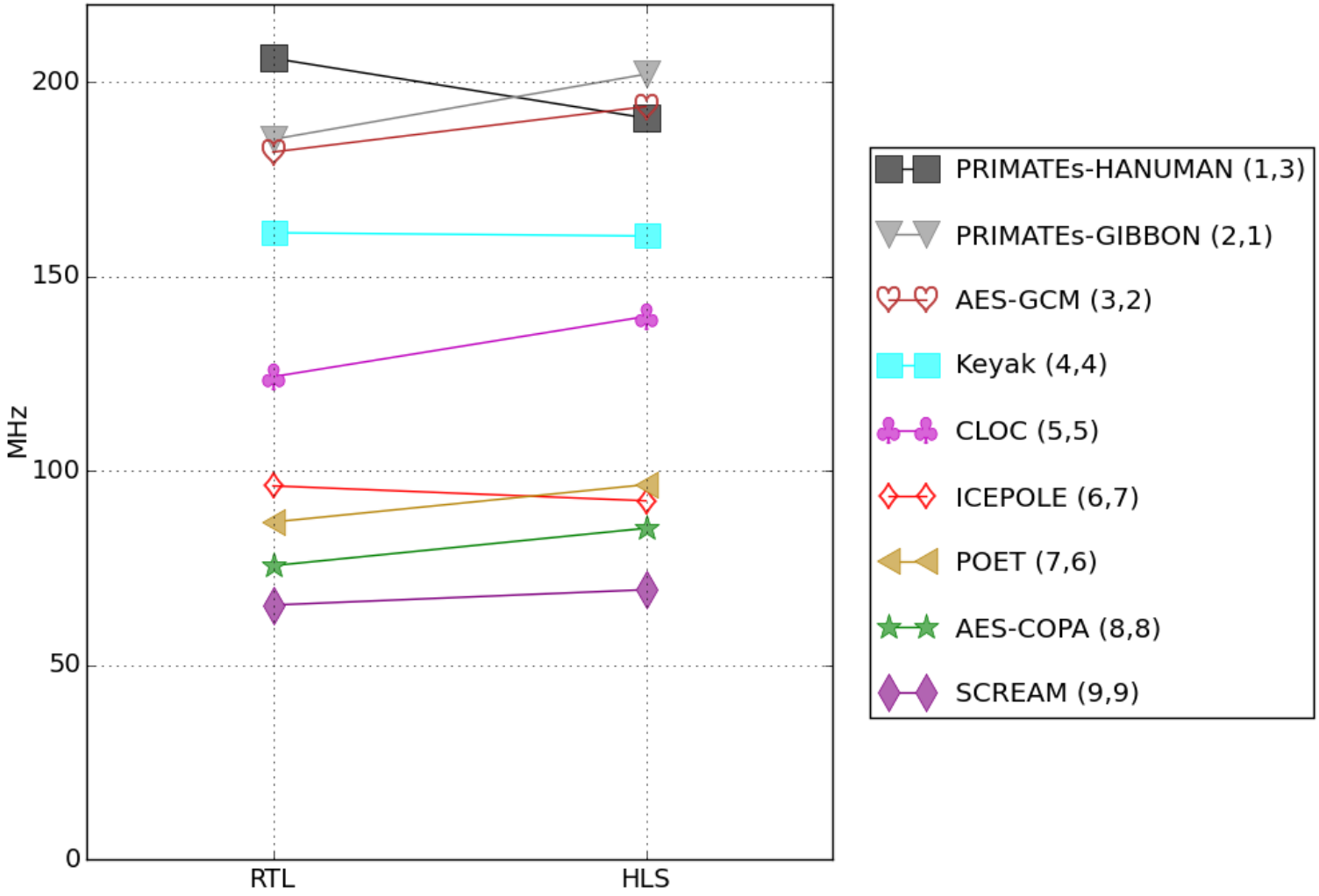
- Difficulty in inferring an overlap between completing the last round and reading the next input block
- One additional clock cycle used for initialization of the state at the beginning of each round
- The formulas for throughput:

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

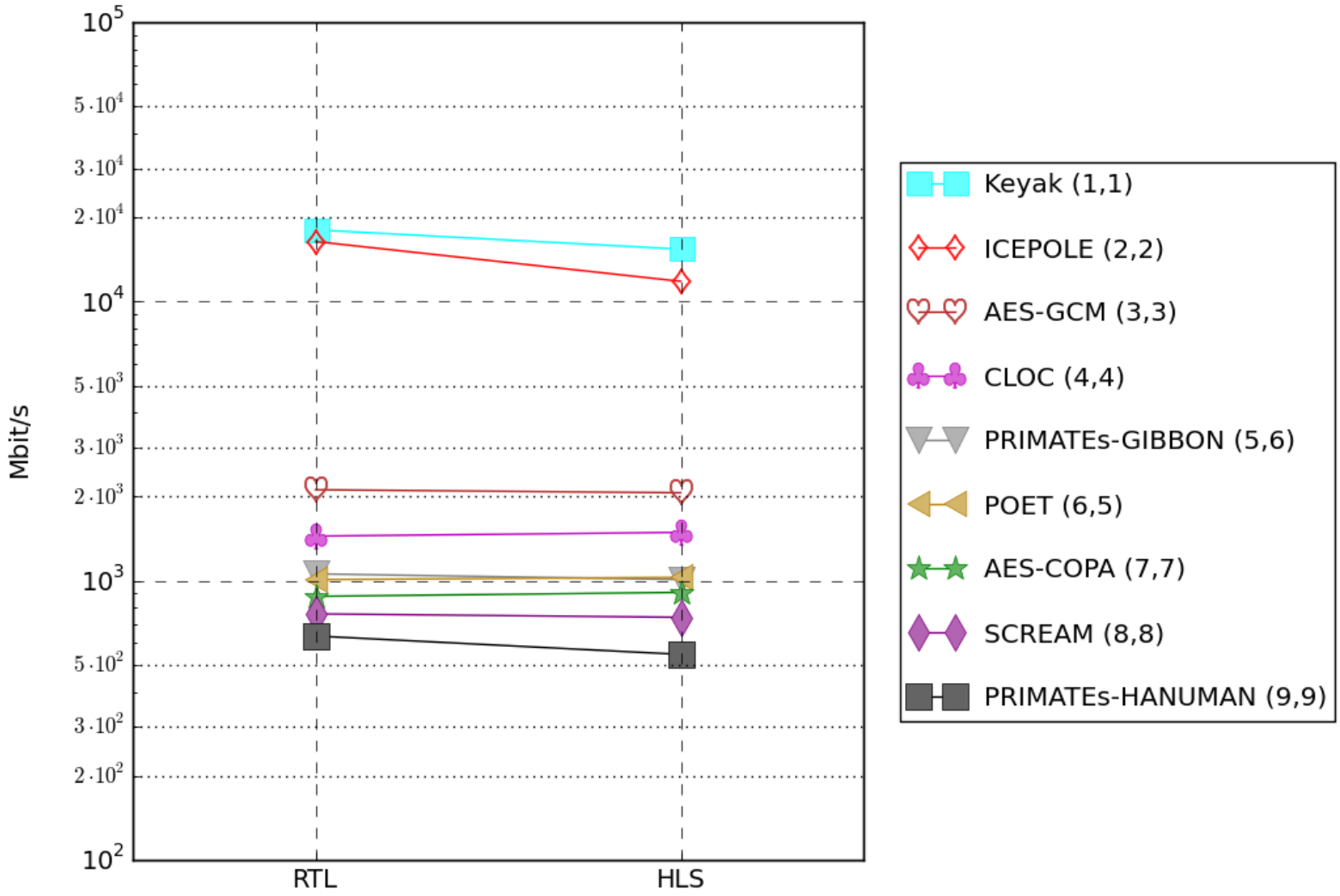
RTL: Throughput = Block_size / (#Rounds+C * T_{CLK})

C=0, 1 depending on the algorithm

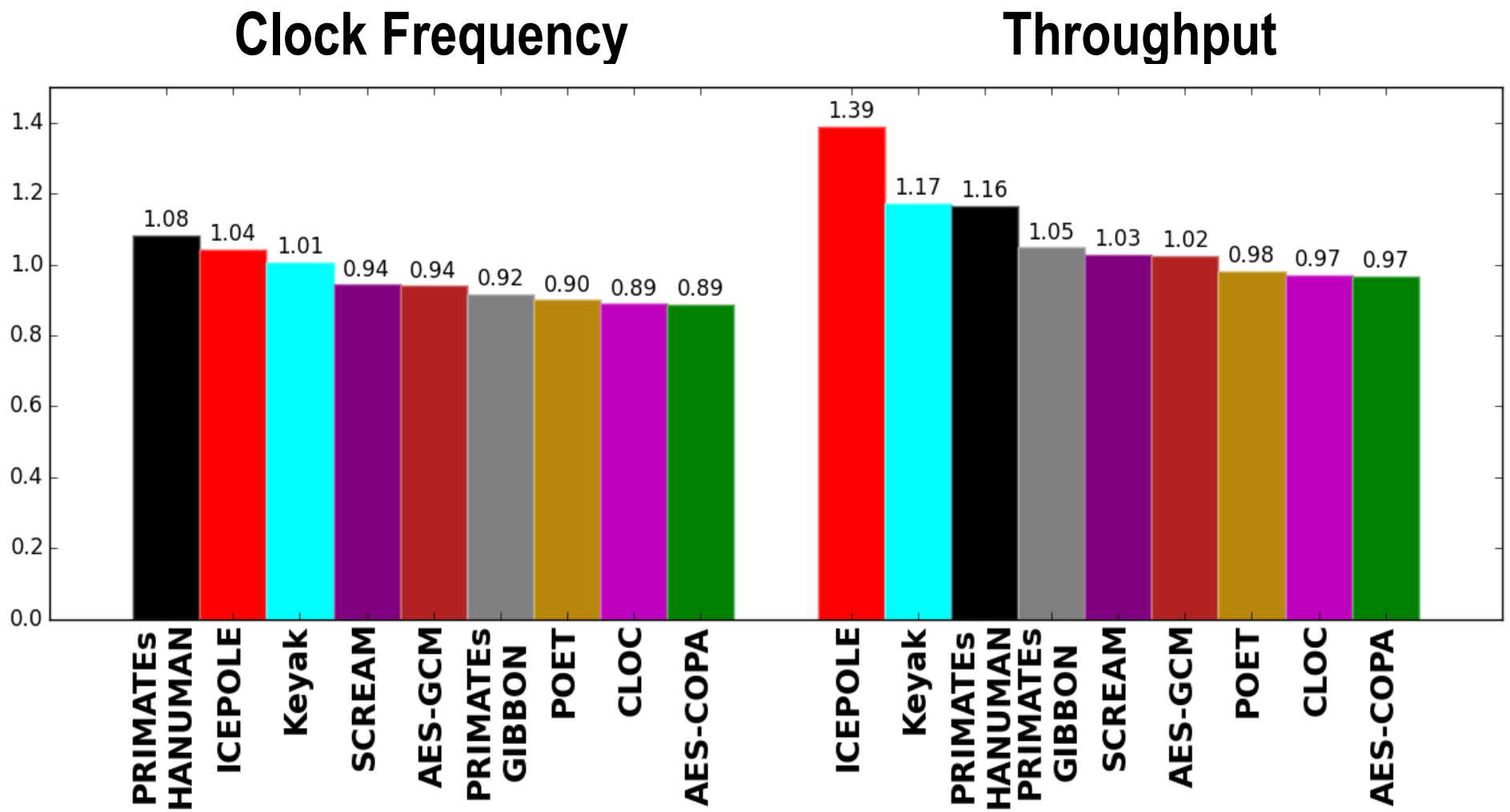
RTL vs. HLS Clock Frequency in Zynq 7000



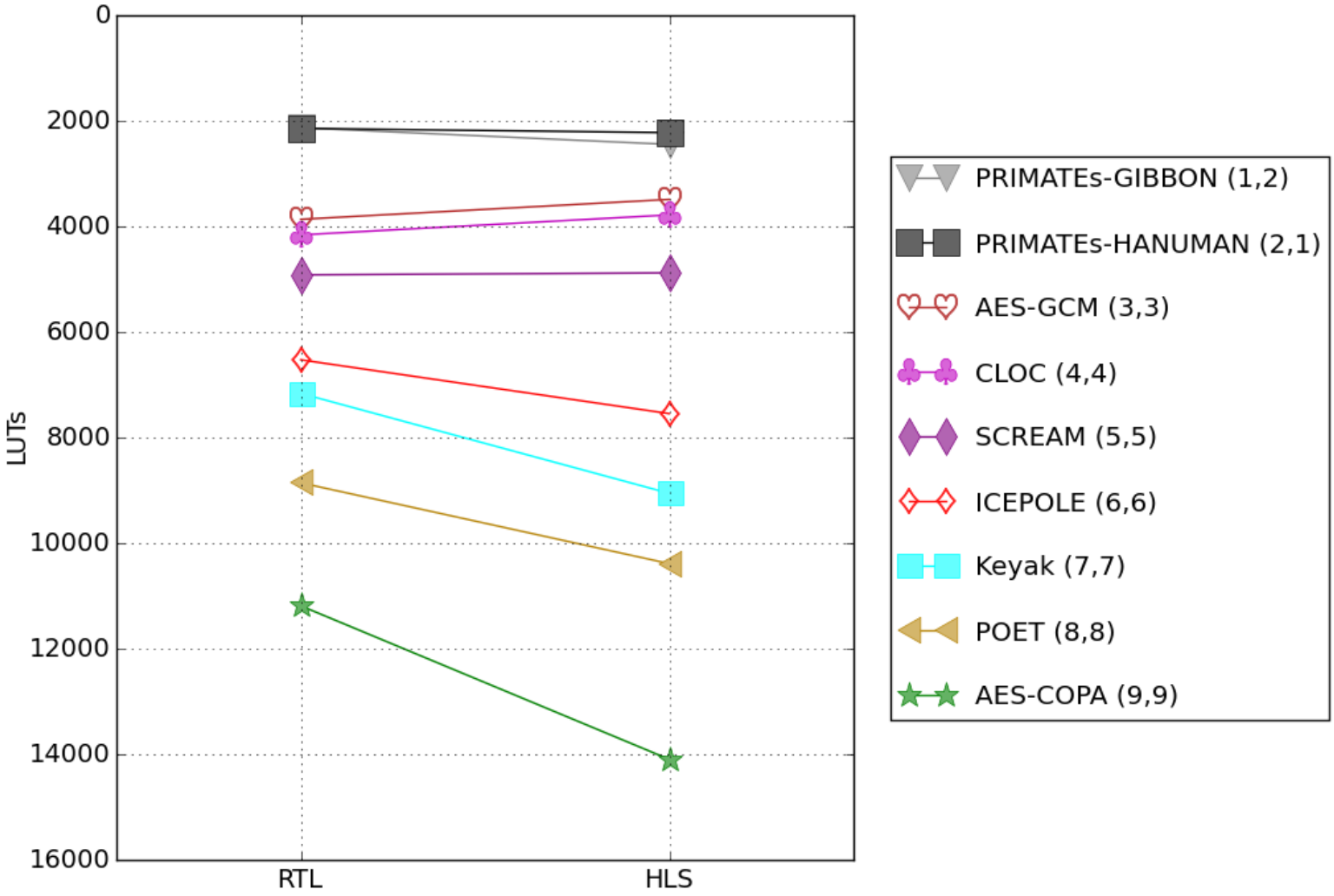
RTL vs. HLS Throughput in Zynq 7000



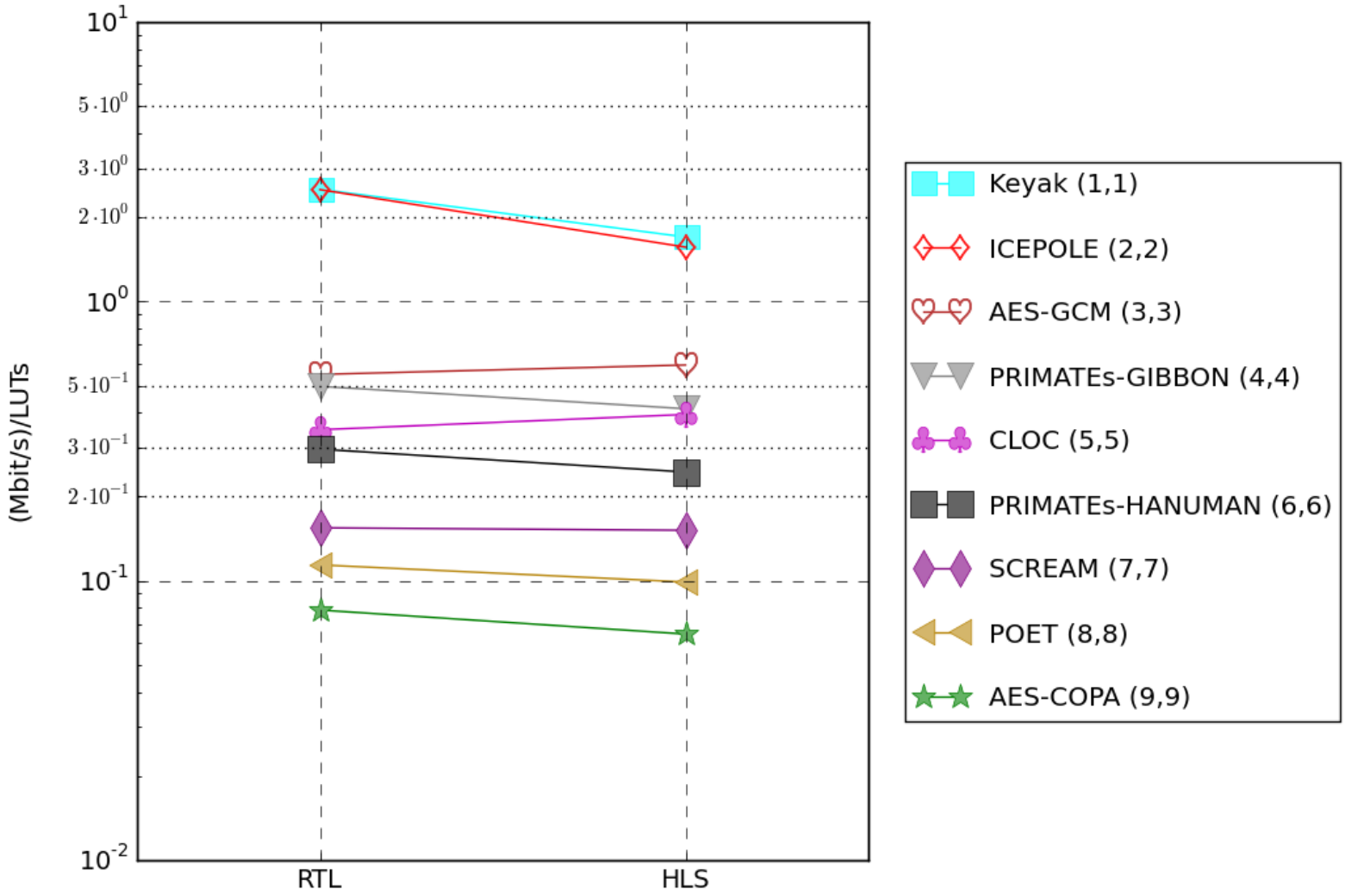
RTL vs. HLS Ratios in Zynq 7000



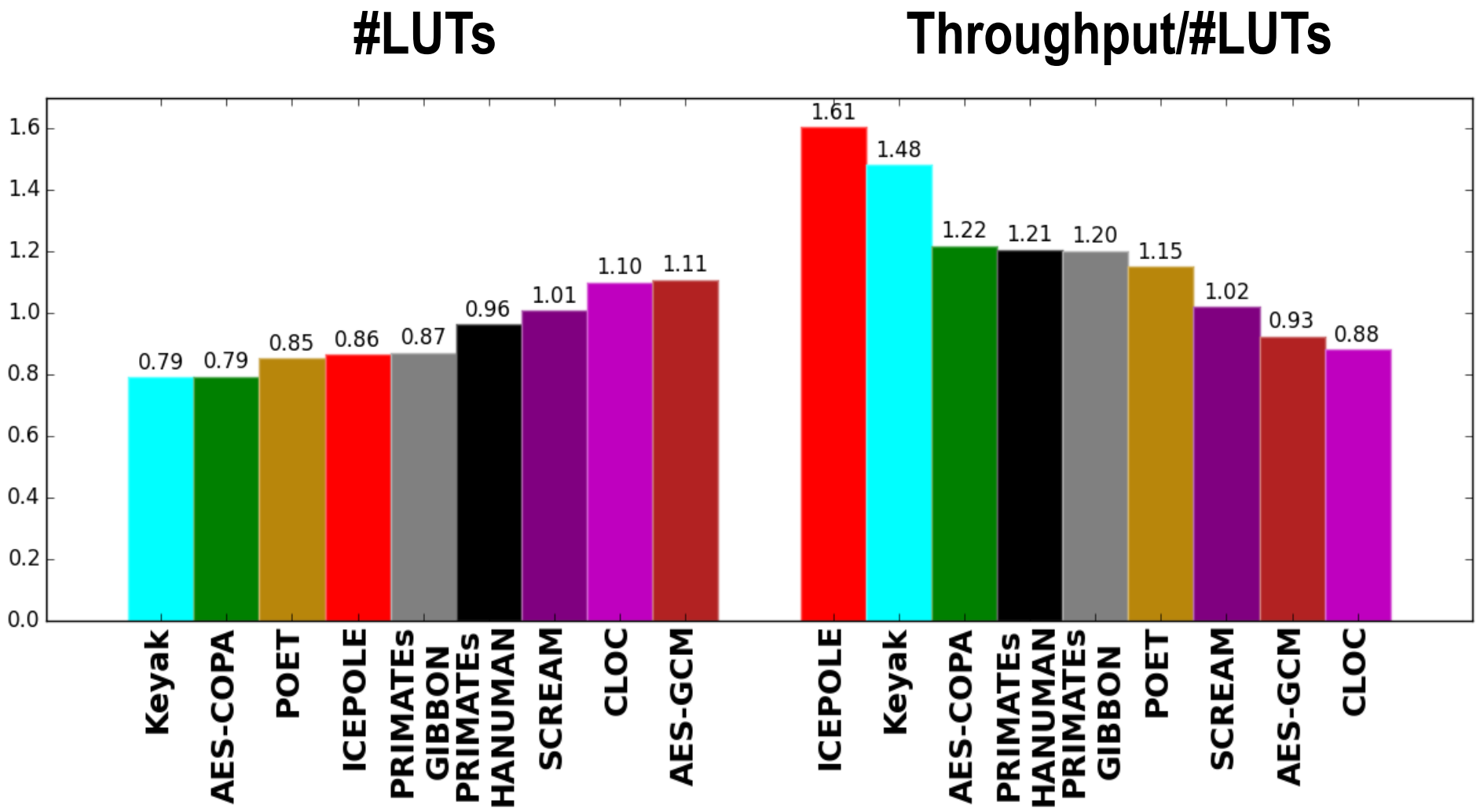
RTL vs. HLS #LUTs in Zynq 7000



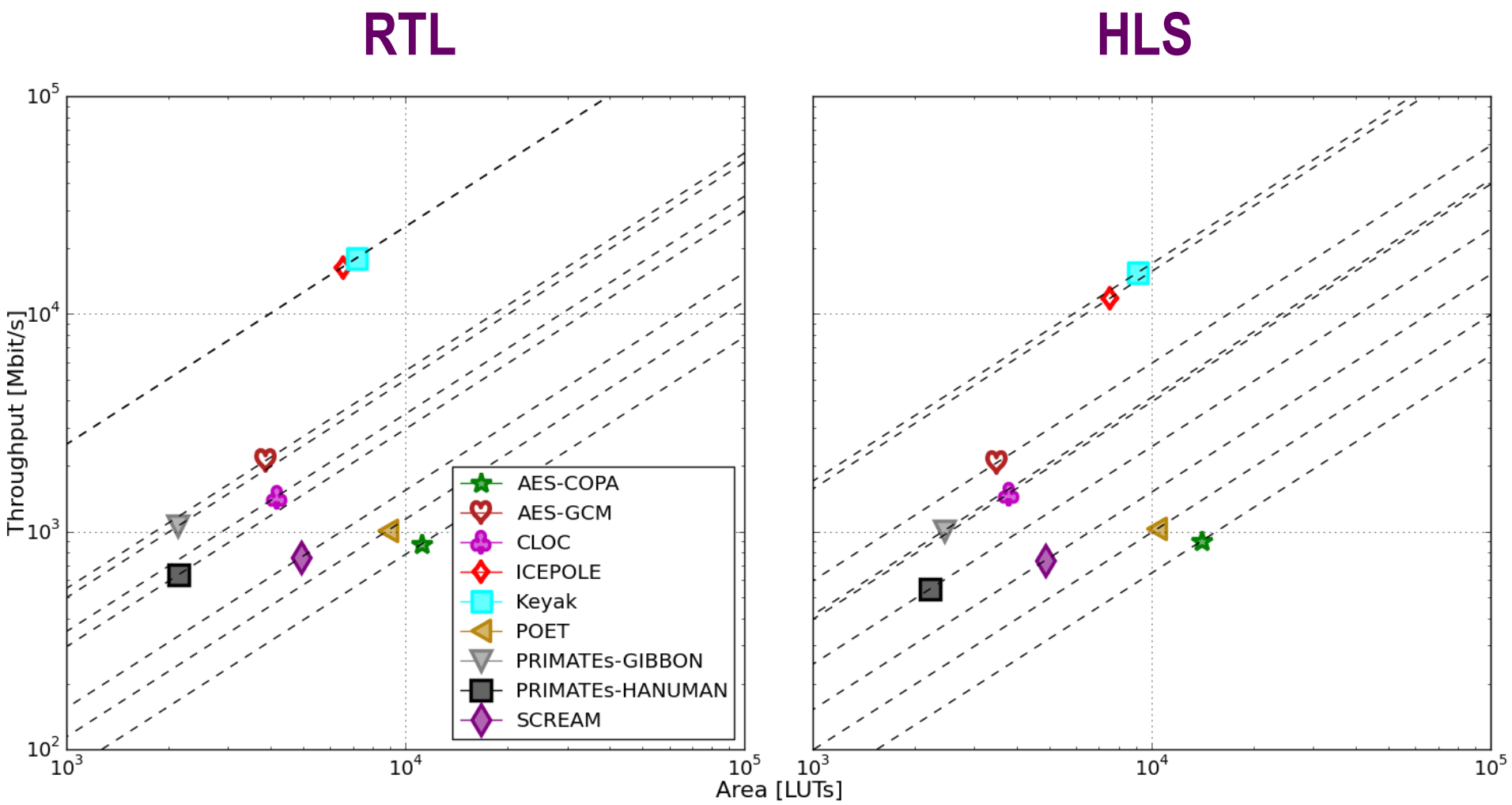
RTL vs. HLS Throughput/#LUTs in Zynq 7000



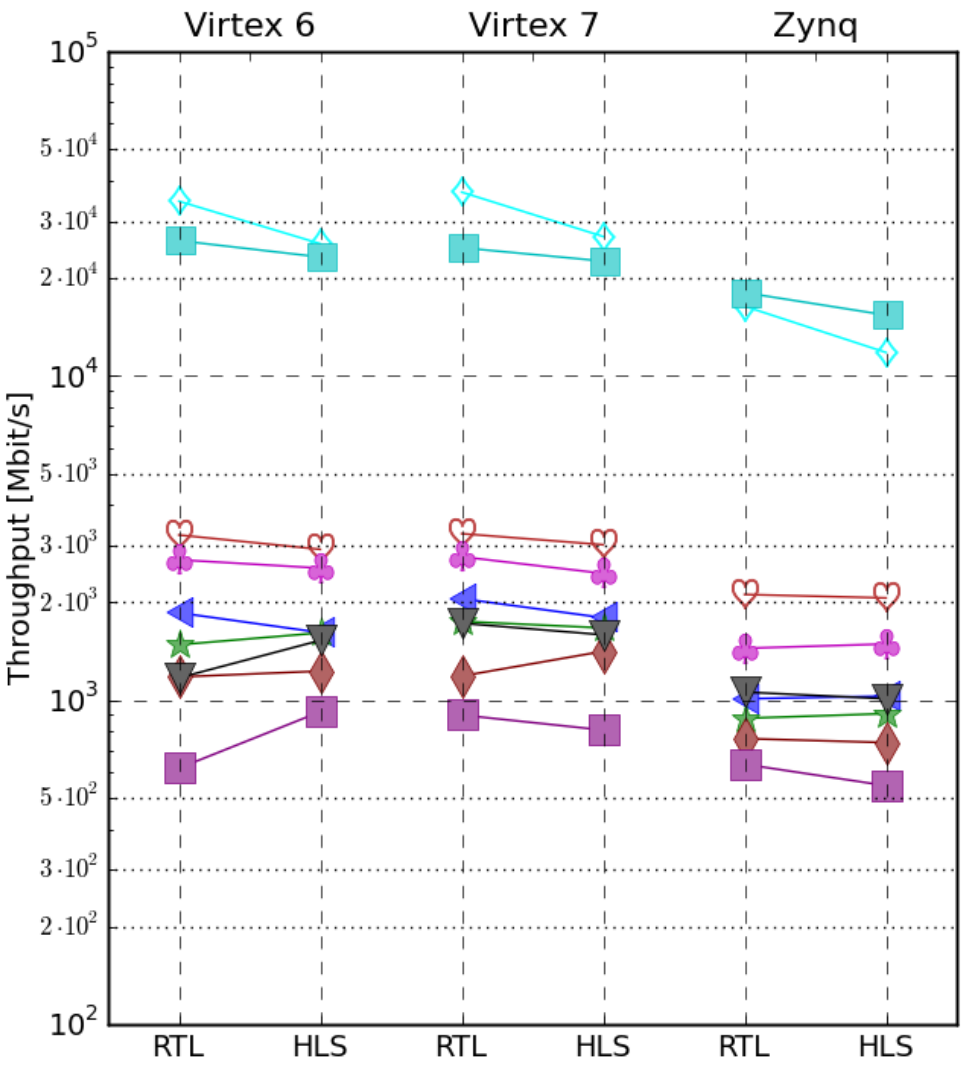
RTL vs. HLS Ratios in Zynq 7000



Throughput vs. LUTs in Zynq 7000

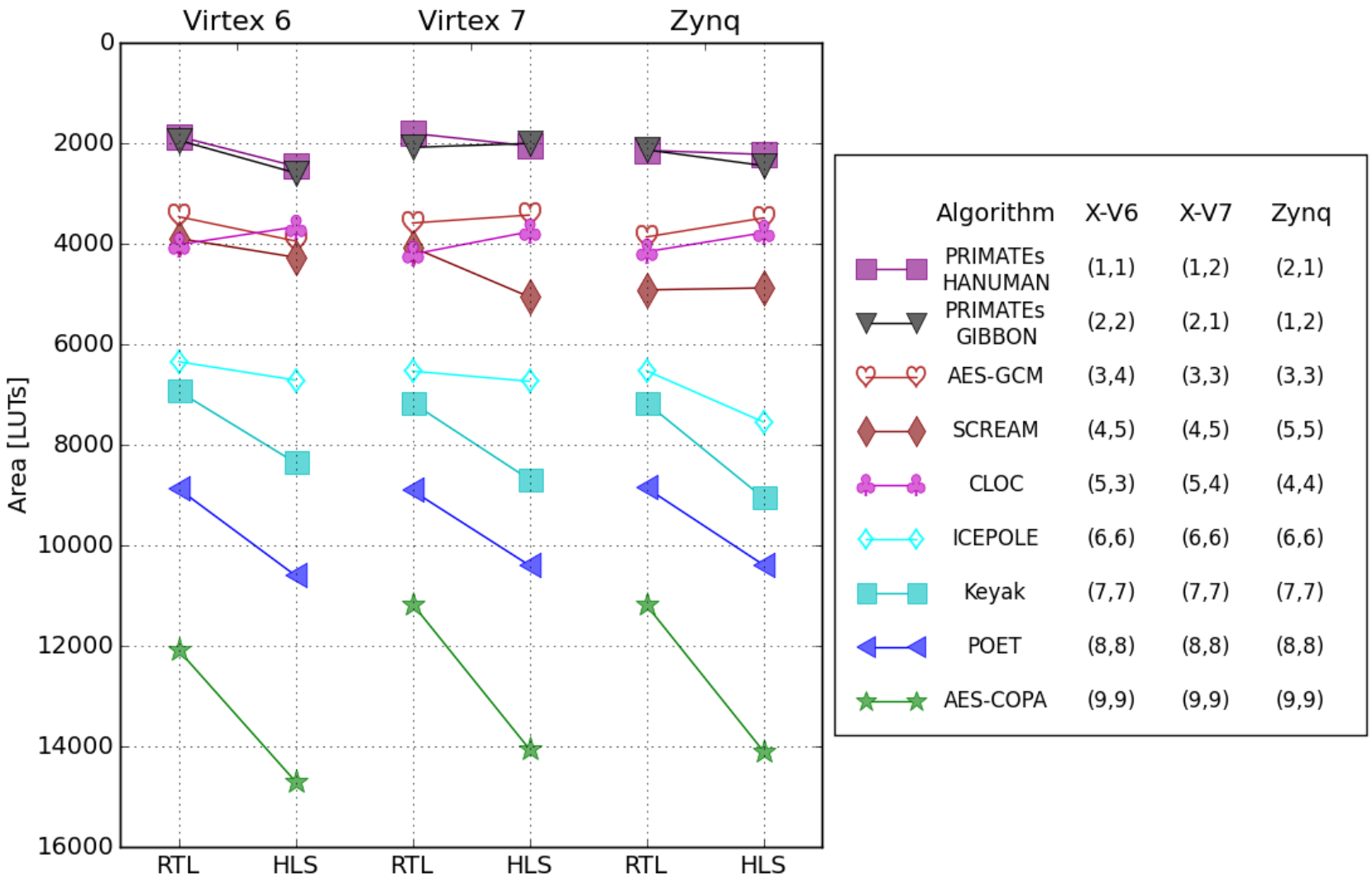


RTL vs. HLS Throughput

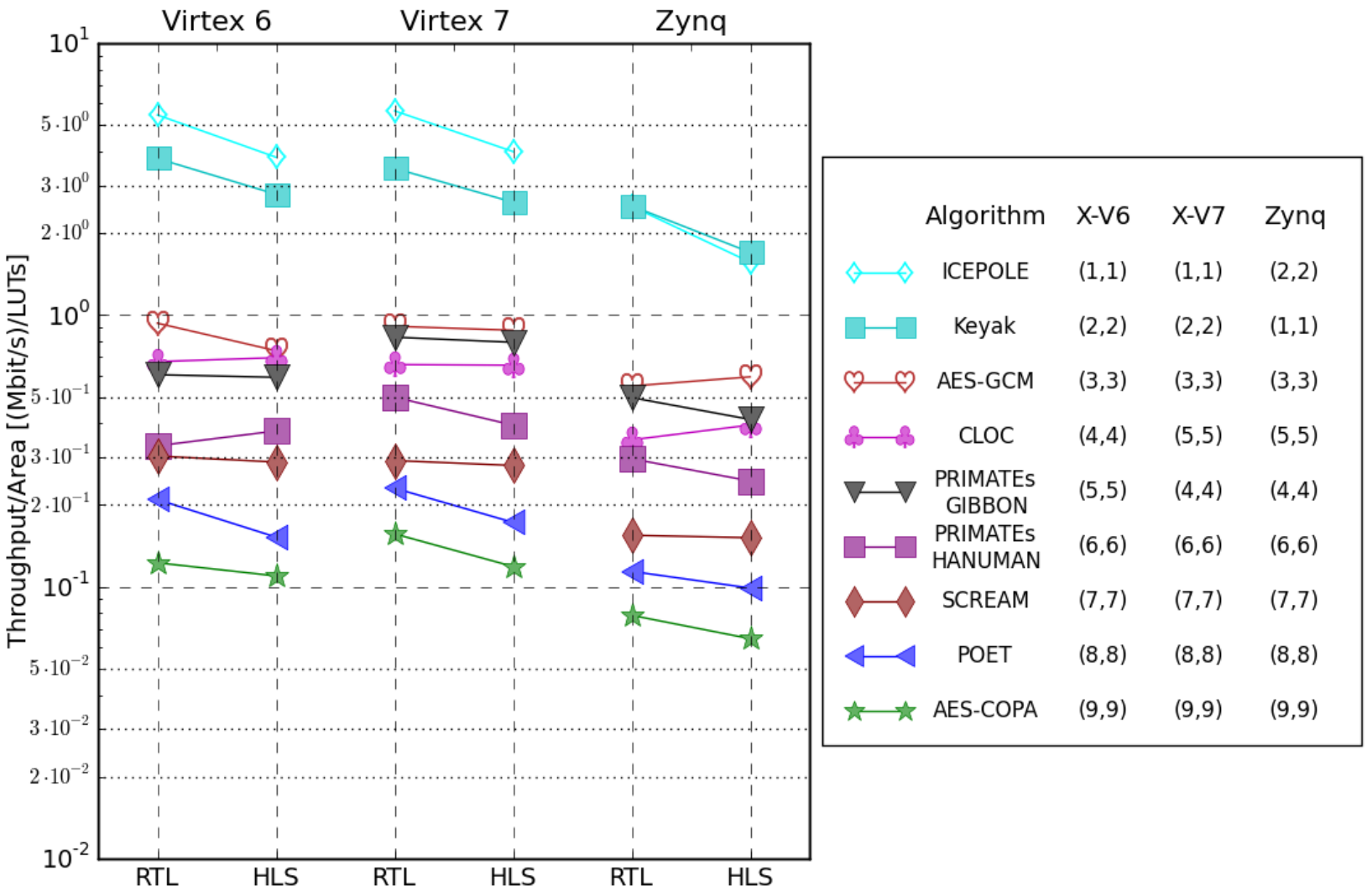


Algorithm	X-V6	X-V7	Zynq
ICEPOLE	(1,1)	(1,1)	(2,2)
Keyak	(2,2)	(2,2)	(1,1)
AES-GCM	(3,3)	(3,3)	(3,3)
CLOC	(4,4)	(4,4)	(4,4)
POET	(5,5)	(5,5)	(6,5)
AES-COPA	(6,6)	(6,6)	(7,7)
SCREAM	(7,8)	(8,8)	(8,8)
PRIMATEs GIBBON	(8,7)	(7,7)	(5,6)
PRIMATEs HANUMAN	(9,9)	(9,9)	(9,9)

RTL vs. HLS #LUTs



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 me regarding an account.
- The ATHENa Option Optimization Tool supports automatic generation of results suitable for uploading to the database

Ordered Listing with a Single-Best (Unique) Result per Each Algorithm

Database of FPGA Results for Authenticated Ciphers

Show Help

Compare Selected

Show 25 entries

About
All FPGA Results
FPGA Rankings
Login

	Algorithm		Design	Platform	Timing
Result ID	Algorithm (Disable Unique)	Key Size [bits]	Implementation Approach	Family	Enc/Auth TP [Mbits/s]
72	ICEPOLE	128	HLS	Virtex 7	26,902
107	Keyak	128	HLS	Virtex 7	22,594
97	AES-GCM	128	HLS	Virtex 7	3,015
101	CLOC	128	HLS	Virtex 7	2,459
111	POET	128	HLS	Virtex 7	1,795
93	AES-COPA	128	HLS	Virtex 7	1,670
116	PRIMATEs-GIBBON	120	HLS	Virtex 7	1,590
89	SCREAM	128	HLS	Virtex 7	1,414
121	PRIMATEs-HANUMAN	120	HLS	Virtex 7	809
Result ID	Algorithm	Key Size [bits]	HLS	Virtex 7	Enc/Auth TP [Mbits/s]

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	96
Transformation Category:	Cryptographic	Cryptographic
Transformation:	Authenticated Cipher	Authenticated Cipher
Group:	Standards	Standards
Algorithm:	AES-GCM	AES-GCM
Tag Size [bits]:	128	128
Associated Data Support:		
Key Size [bits]:	128	128
Secret Message Number:		
Secret Message Number Size [bits]:	-	-
Message Block Size [bits]:	128	128
Other Parameters:		
Specification:	**SP-800-38D.pdf**	**SP-800-38D.pdf**
Formula for Message Size After Padding:		
Design		
Design ID:	20	21
Impl Approach:	RTL	HLS
Hardware API:	GMU_AEAD_Core_API_v1	GMU_AEAD_Core_API_v1
Primary Optimization Target:	Throughput/Area	Throughput/Area
Secondary Optimization Target:		
Architecture Type:	Basic Iterative	Basic Iterative
Description Language:	VHDL	VHDL
Use of Megafunctions or Primitives:	No	No
List of Megafunctions or Primitives:		
Maximum Number of Streams Processed in Parallel:	1	1
Number of Clock Cycles per Message Block in a Long Message:	11	12
Datapath Width [bits]:	128	128
Padding:	Yes	Yes
Minimum Message Unit:		
Input Bus Width [bits]:	32	32

Comparison of Result #s 95 and 97		
Platform		
Device Vendor:	Xilinx	Xilinx
Family:	Virtex 7	Virtex 7
Device:	xc7vx485tffg1761-2	xc7vx485tffg1761-2
Timing		
Encryption/Authentication Throughput [Mbits/s]:	3261	3015
Decryption/Authentication Throughput [Mbits/s]:	3261	3015
Authentication-Only Throughput [Mbits/s]:	3261	3015
Synthesis Clock Frequency [MHz]:	-	-
Key Scheduling Time [ns]:	-	-
Requested Synthesis Clock Frequency [MHz]:	-	-
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		
CLB Slices:	1166	1105
LUTs:	3588	3430
Flip Flops:	-	-
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 8 Round 1 CAESAR candidates & AES-GCM demonstrated correct ranking for majority of candidates using all major performance metrics

- More research needed to overcome remaining difficulties
    - Suboptimal control unit
    - Wide range of RTL to HLS performance metric ratios
    - 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