An Alternative Approach to Hardware Benchmarking of CAESAR Candidates Based on the Use of High-Level Synthesis Tools

Ekawat Homsirikamol
and Kris Gaj
George Mason University
USA

Based on work partially supported by the National Science Foundation under Grant No. 1314540

First Author

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”

Number of Candidates in Cryptographic Contests

	Initial number of candidates	Implemented in hardware	Percentage
AES	15	5	33.3%
eSTREAM	34	8	23.5%
SHA-3	51	14	27.5%
CAESAR	57	28	49.1%

Pros & Cons of Multiple Designers

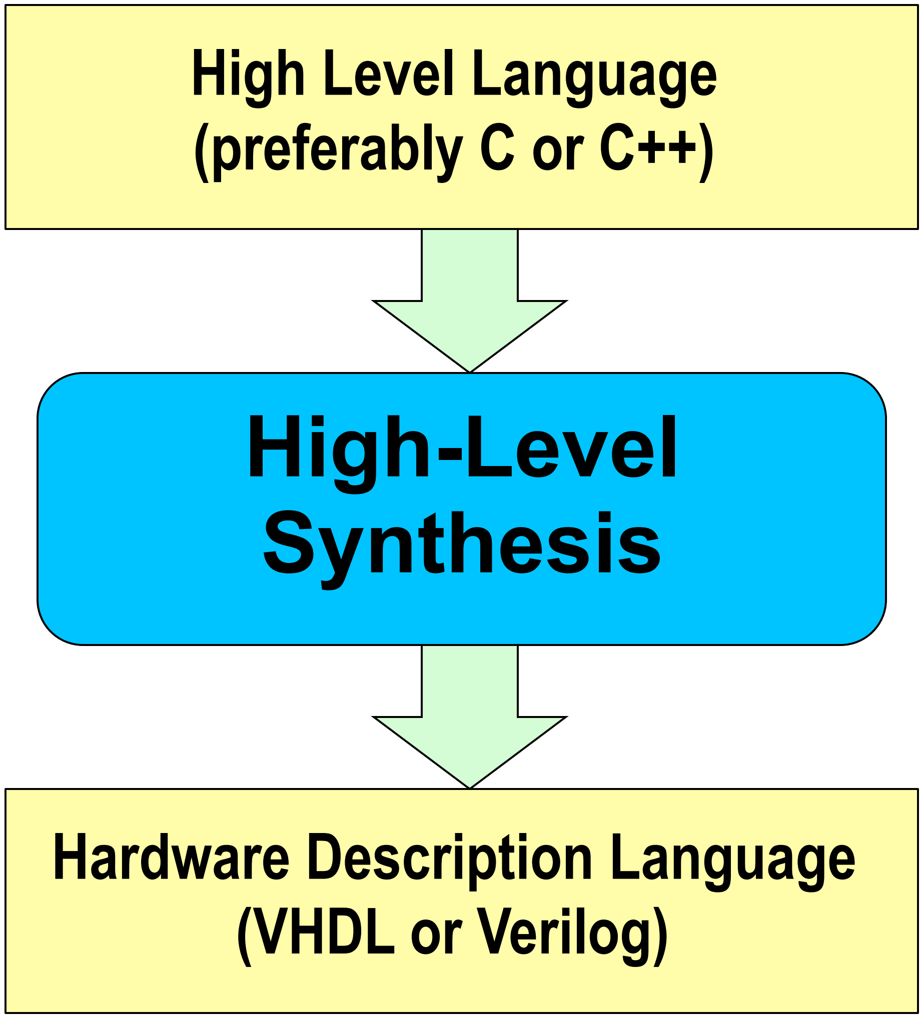
Pros:

- Distribution of effort
- Larger talent pool
- Potential for design space exploration

Cons:

- Different skills of designers
- Different amount of time and effort
- Misunderstandings regarding API and optimization target
- Requests for extending the deadline or disregarding ALL results

Potential Solution: High-Level Synthesis (HLS)



Case for High-Level Synthesis & Crypto

- Each submission includes **reference implementation in C**
- **Development time** potentially **decreased 3-10 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

Potential Additional Benefits

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

Typical Doubts (from reviewers of our papers)

- How can we trust these tools?
- Isn’t manual design always better?
- Is it fair to compare manual designs with HLS designs?
- Won’t the number of candidates saturate soon anyway?

Typical Doubts (from reviewers of our papers)

- How can we trust these tools?
- Isn’t manual design always better?
- Is it fair to compare manual designs with HLS designs?
- Won’t the number of candidates saturate soon anyway?
- Why did not you implement Serpent?

  (the same reviewer at two major crypto conferences)

High-Level Synthesis: State of the Art

"A Survey and Evaluation of FPGA High-Level Synthesis Tools"

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume: 35, Issue: 10, Oct. 2016)

Razvan Nane, Vlad-Mihai Sima, Koen Bertels:
Delft University of Technology, The Netherlands

Christian Pilato, Fabrizio Ferrandi:
Politecnico di Milano, Italy

Jongsok Choi, Blair Fort, Andrew Canis, Yu Ting Chen, Hsuan Hsiao, Stephen Brown, Jason Anderson:
University of Toronto, Canada

Number of Tools

	C, C++, or Extended C	Other Languages
In Use	14	3
Abandoned	7	4
Status Unknown	5	0
Total	26	7

Number of Tools supporting C, C++, Extended C

	Commercial	Academic
In Use	10	4
Abandoned	1 (C2H)	6
Status Unknown	1	4
Total	12	14

In-Use Tools supporting C, C++, Extended C

Commercial:

- 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; Synphony C: Synopsys
- Vivado HLS: Xilinx

Academic:

- Bambu: Politecnico di Milano, Italy
- DWARV: Delft University of Technology, The Netherlands
- GAUT: Universite de Bretagne-Sud, France
- LegUp: University of Toronto, Canada

Crypto-related Benchmarks (C programs)

CHStone Benchmark Program Suite for Practical C-based High-Level Synthesis

http://www.ertl.jp/chstone/

aes-encrypt:
Key scheduling + Encryption of **1** 128-bit **block**

aes-decrypt:
Key scheduling + Decryption of **1** 128-bit **block**

sha:
Hashing of **256** 512-bit **blocks** using SHA-1

blowfish:
Key scheduling + Encryption of **650** 64-bit **blocks** in CFB64 mode

Benchmarking Results in Number of Clock Cycles Before Optimization

Tools	aes-encrypt	aes-decrypt	sha	blowfish
Bambu	1,574	2,766	111,762	57,590
DWARV	5,135	2,579	71,163	70,200
LegUp	1,564	7,367	168,886	75,010
Commercial	3,976	5,461	197,867	101,010
Manual	20	20	20,480	18,736
Best/Manual	78	129	3.5	3.1

Benchmarking Results in Number of Clock Cycles After Optimization

Tools	aes-encrypt	aes-decrypt	sha	blowfish
Bambu	1,485	2,585	51,399	57,590
DWARV	3,282	2,579	71,163	70,200
LegUp	1,191	4,847	81,786	64,480
Commercial	3,735	3,923	124,339	96,460
Manual	20	20	20,480	18,736
Best/Manual	60	129	2.5	3.1

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**
  - **8 out of 8**: Operation Chaining, Bitwidth Analysis and Optimization, Memory Space Allocation, Loop Optimizations, Hardware Resource library, Speculation and Code Motion, If-Conversion [**Bambu, LegUp: 6 out of 8, DWARV: 5 out of** 8]
- On average the **highest clock frequency** of the generated code

Licensing Limitations of Vivado HLS

1. Results cannot be compared with results obtained using other HLS tools
2. Designers are not allowed to target ASICs
3. Designers are not allowed to target devices of other FPGA vendors (e.g., Altera)

GMU (Ice's) Previous Efforts (1)

AES-128-ECB-ENC (Spartan 6):
ReConFig (Reconfigurable Computing and FPGAs), Dec. 2014

HLS/RTL ratios:

• Clock cycles:	12/10 = 1.2
• Area:	343/354 = 0.97

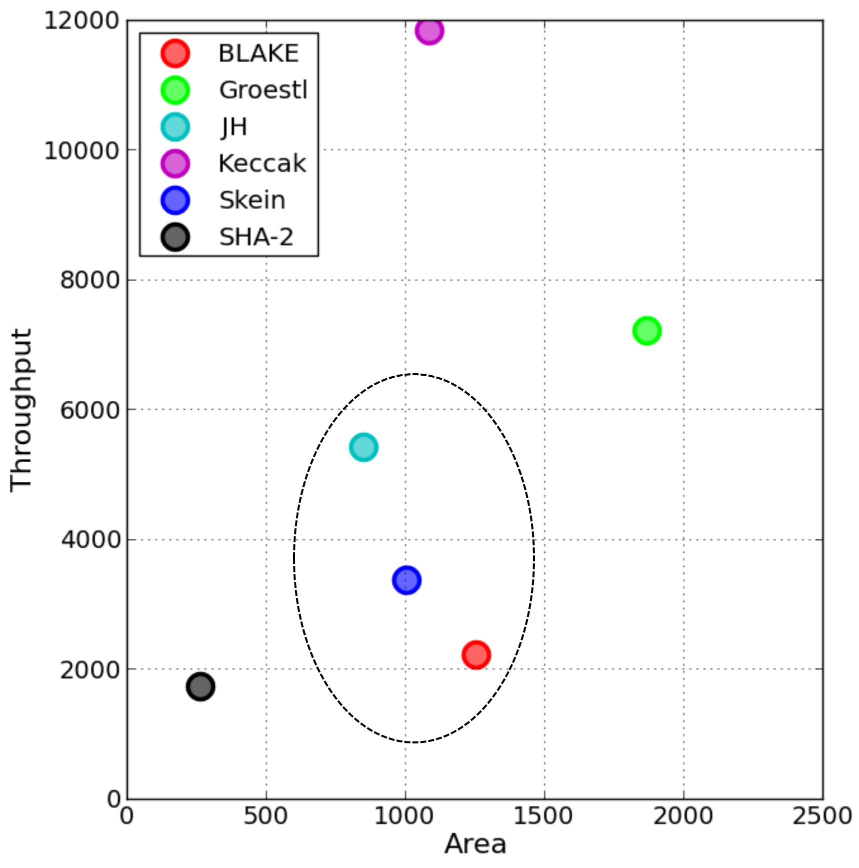
RTL/HLS ratios:

• Frequency:	230/231 = 0.996
• Throughput:	2943/2467 = 1.19
• Throughput/Area:	8.31/7.19 = 1.16

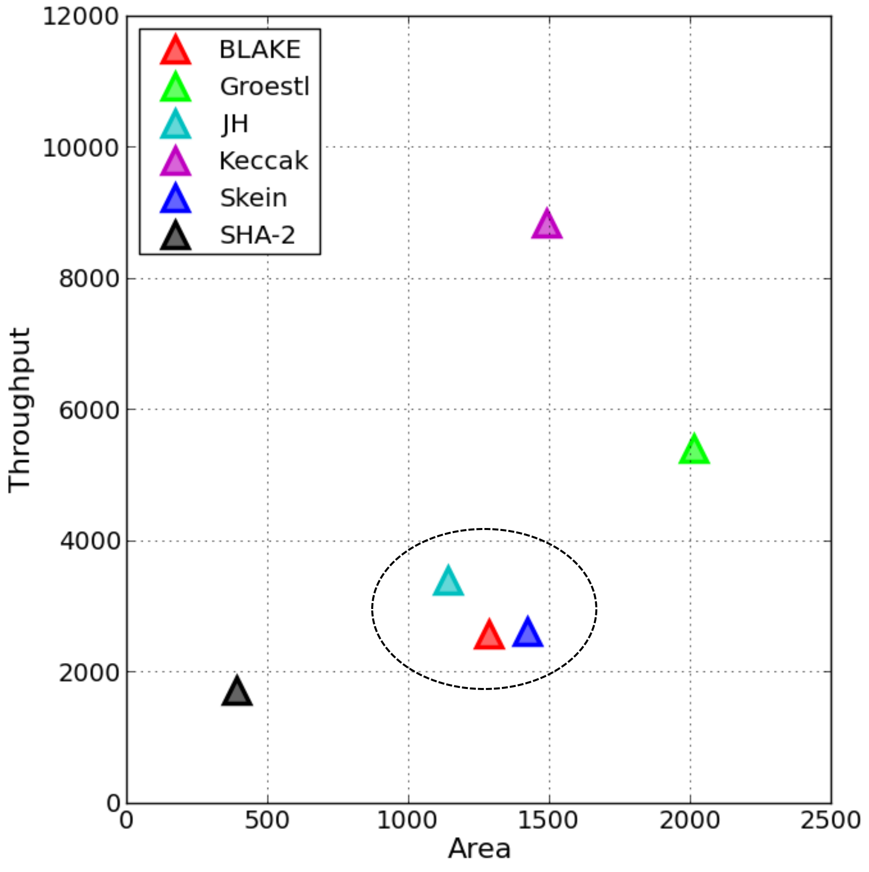
GMU (Ice's) Previous Efforts (2)

5 Final SHA-3 Candidates & SHA-2 (Virtex 6):

ARC (Applied ReConfigurable Computing, Apr. 2015



RTL



HLS

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?**

18 months of unsuccessful publishing attempts and unread/ignored rebuttals

1. Why not other HLS tools ?
2. Why not ASICs ?
3. Why not other FPGA vendors (e.g., Altera)?
4. Why no previous work by other teams?
5. Why another publication?

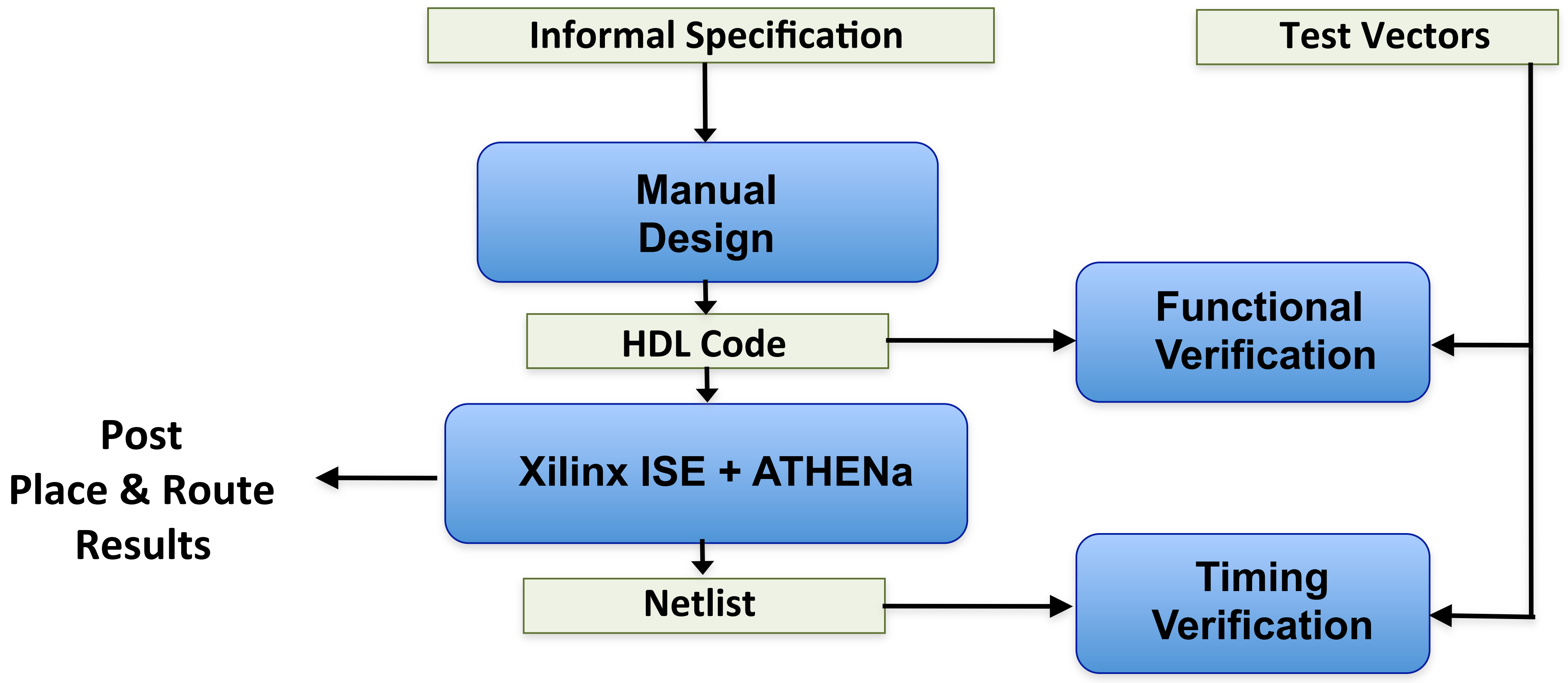
18 months of unsuccessful publishing attempts and unread/ignored rebuttals

1. Why not other HLS tools ?
2. Why not ASICs ?
3. Why not other FPGA vendors (e.g., Altera)?
4. Why no previous work by other teams?
5. Why another publication?
6. Why not Serpent?

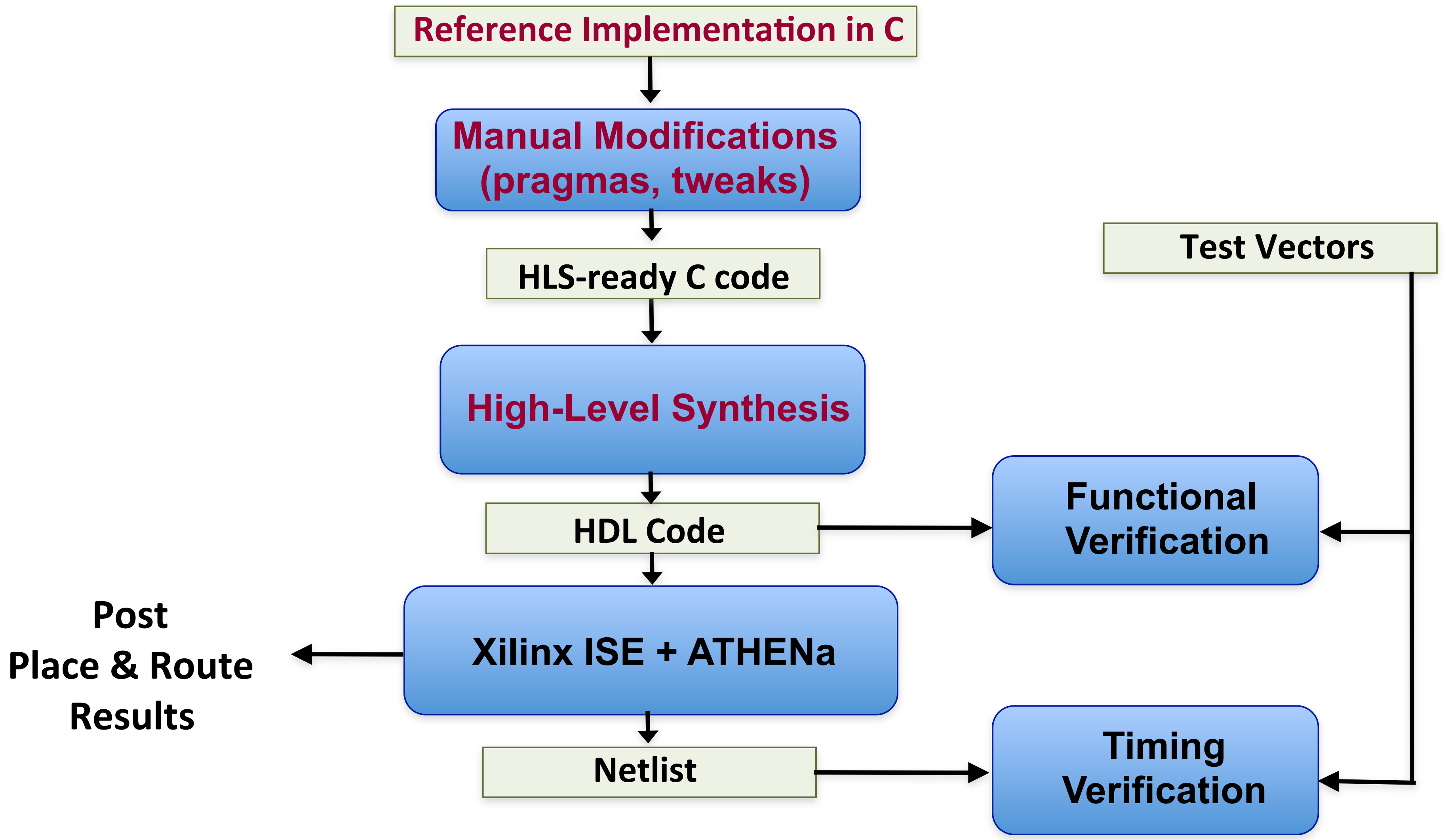
DIAC 2016 vs. DIAC 2015

- **CAESAR HW API 1.0** (02/2016) vs. GMU API 1.1 (09/2015)
- Comparison vs. **RTL implementations** developed **by other groups**
- **New candidates** (e.g., MORUS, AEGIS, NORX, SILC)
- Block-based => **stream-based implementation**
- Easily **adjustable** algorithm-dependent **port widths**
- **C++ testbench independent of hardware architecture**
- Automated generation of **test vectors at the CipherCore (C++) level**

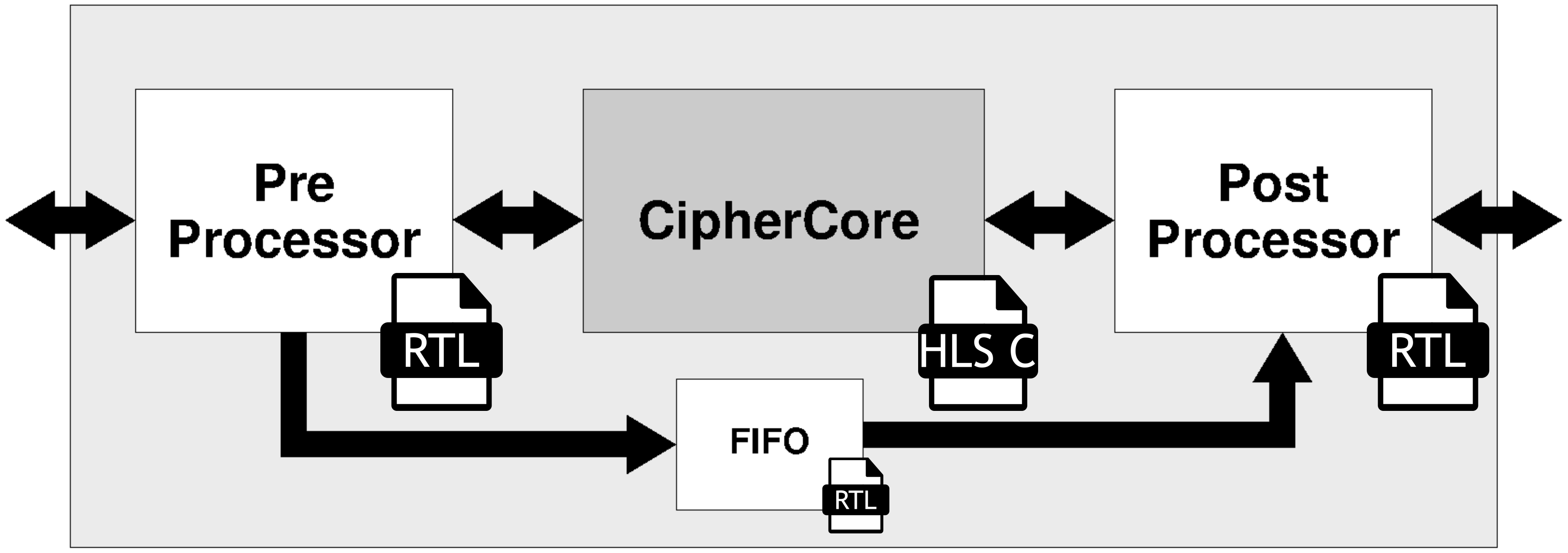
Traditional Register-Transfer Level (RTL) Development & Benchmarking Flow



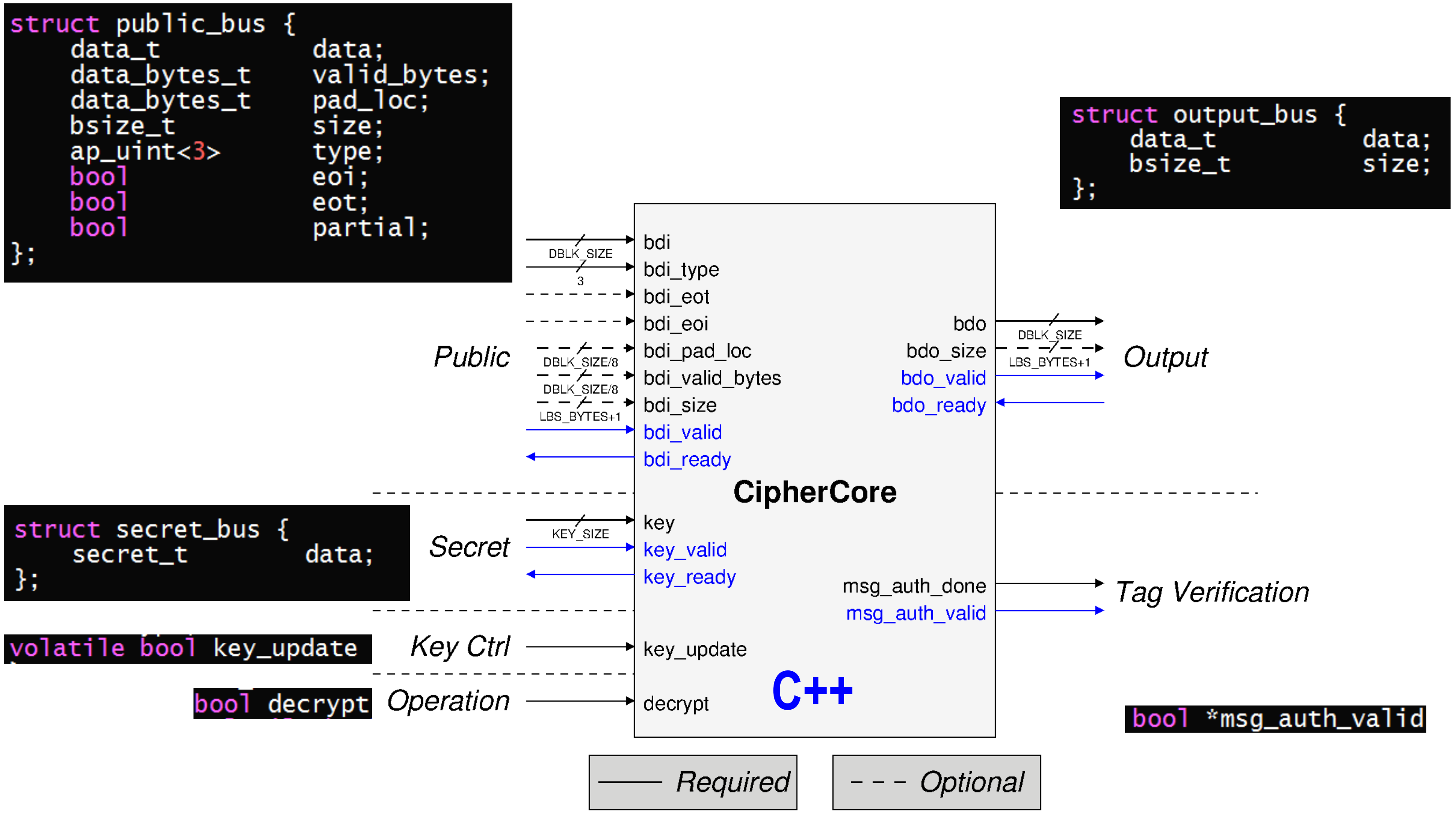
Proposed HLS-Based Development and Benchmarking Flow



Language Partitioning



Mapping Hardware to Software Interface



```
struct public_bus {
    data_t          data;
    data_bytes_t    valid_bytes;
    data_bytes_t    pad_loc;
    bsize_t         size;
    ap_uint<3>      type;
    bool            eoi;
    bool            eot;
    bool            partial;
};
```

```
struct output_bus {
    data_t          data;
    bsize_t         size;
};
```

```
struct secret_bus {
    secret_t        data;
};
```

```
volatile bool key_update
```

```
bool decrypt
```

```
bool *msg_auth_valid
```

Basic handshaking signals (valid, ready) added automatically

Easily Adjustable Port Widths

```
struct public_bus {
    data_t          data;
    data_bytes_t    valid_bytes;
    data_bytes_t    pad_loc;
    bsize_t         size;
    ap_uint<3>      type;
    bool            eoi;
    bool            eot;
    bool            partial;
};
```

```
struct secret_bus {
    secret_t        data;
};
```

```
struct output_bus {
    data_t          data;
    bsize_t         size;
};
```

```
/* Interface data types */
typedef ap_uint<KEYBUS>       secret_t;
typedef ap_uint<BLOCKSIZE>    data_t;
typedef ap_uint<BLOCKSIZE/8>  data_bytes_t;
typedef ap_uint<LBS_BYTES+1>  bsize_t;
```

```
/* Interface parameters */
#define KEYSIZE 128
#define KEYBUS 32
#define LBS_BYTES 4
```

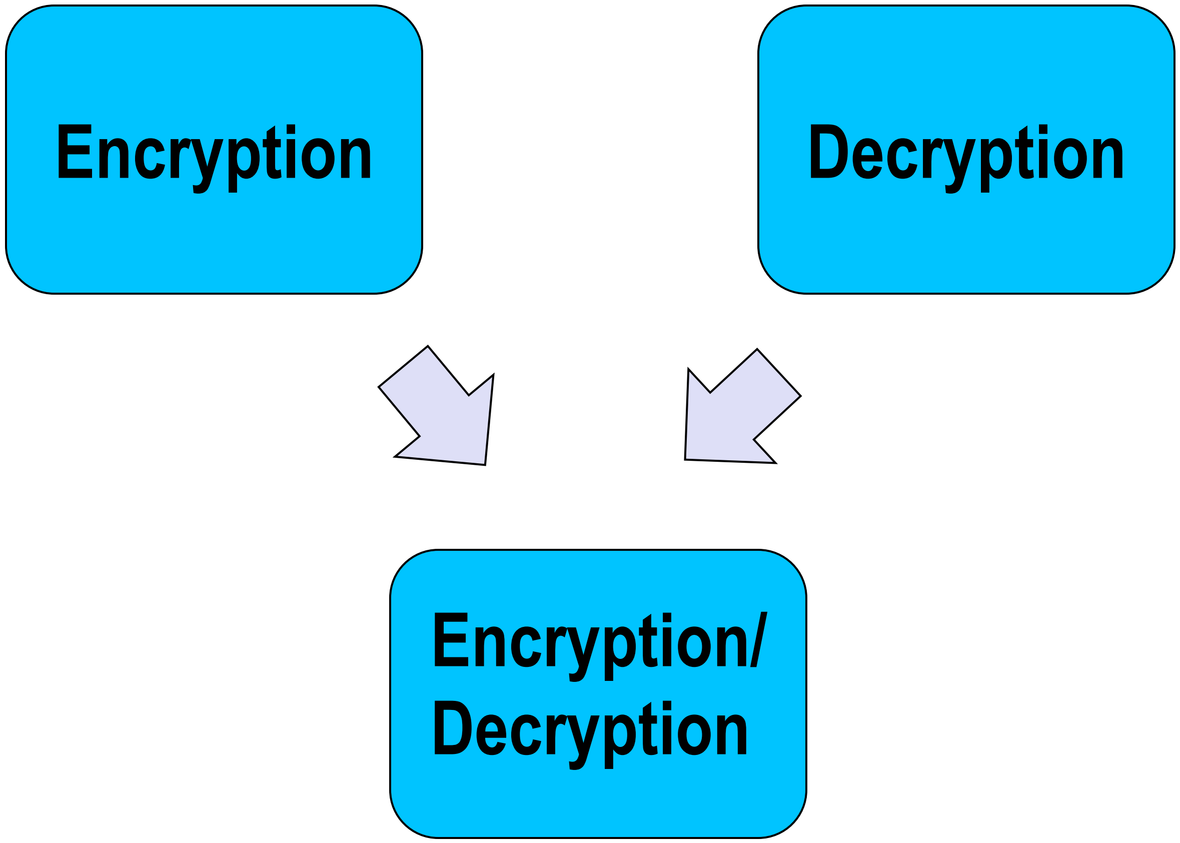
Reference C vs. HLS-ready C/C++

Data	Reference C	HLS-ready C/C++
Access	Random Data can be accessed at any location multiple times	Serial Previously accessed data must be maintained inside of the code if required
Width	Byte/Word	Block size
Total Size	Known	Unknown
Status	Always available	Availability unknown until the time of read

Reference C vs. HLS-ready C/C++

Reference C

HLS-ready C/C++



Use of pragmas possible but unreliable

Low-Level Code Rewriting

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);
  }
```

Adding Pragmas

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
            ...
}
```

Change array shapes:

```
void AES_encrypt (word8 a[4][4], word8 k[4][4], word8 b[4][4])
{
#pragma HLS ARRAY_RESHAPE variable=a[0] complete dim=1 reshape
#pragma HLS ARRAY_RESHAPE variable=a[1] complete dim=1 reshape
#pragma HLS ARRAY_RESHAPE variable=a[2] complete dim=1 reshape
#pragma HLS ARRAY_RESHAPE variable=a[3] complete dim=1 reshape
#pragma HLS ARRAY_RESHAPE variable=a complete dim =1 reshape
```

HLS-Ready C/C++ Code Generation

Phase I

1. Step-by-step **designer's guide** (**under development**)
   - Code rewriting
   - Pragmas insertion
2. Multiple **examples** (AES, SHA-3, CAESAR contests)

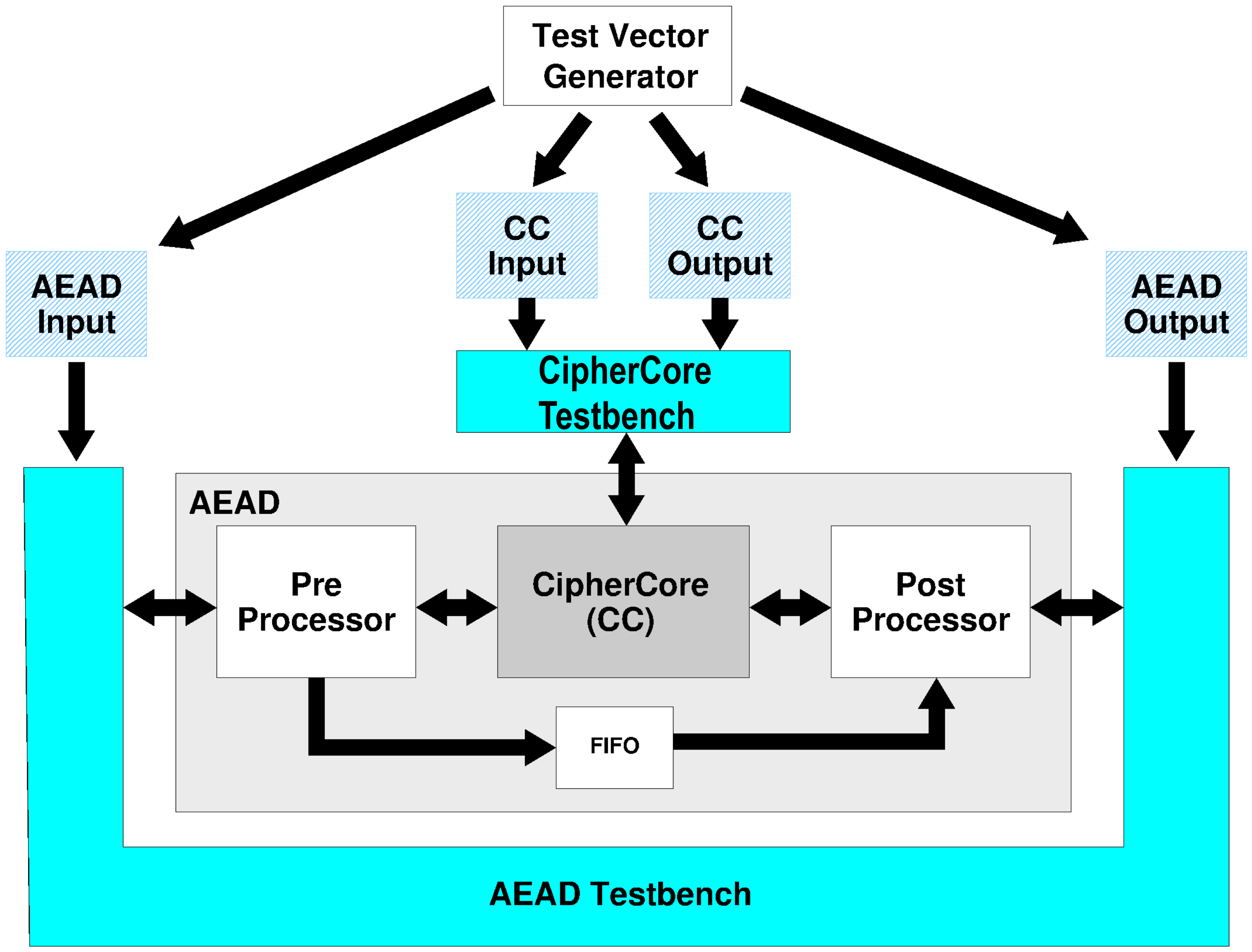
Phase II

1. **Automated insertion of pragmas** for Vivado HLS
2. **Translation of** Vivado HLS **pragmas** to pragmas for academic tools: Bambu, DWARV, LegUp

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

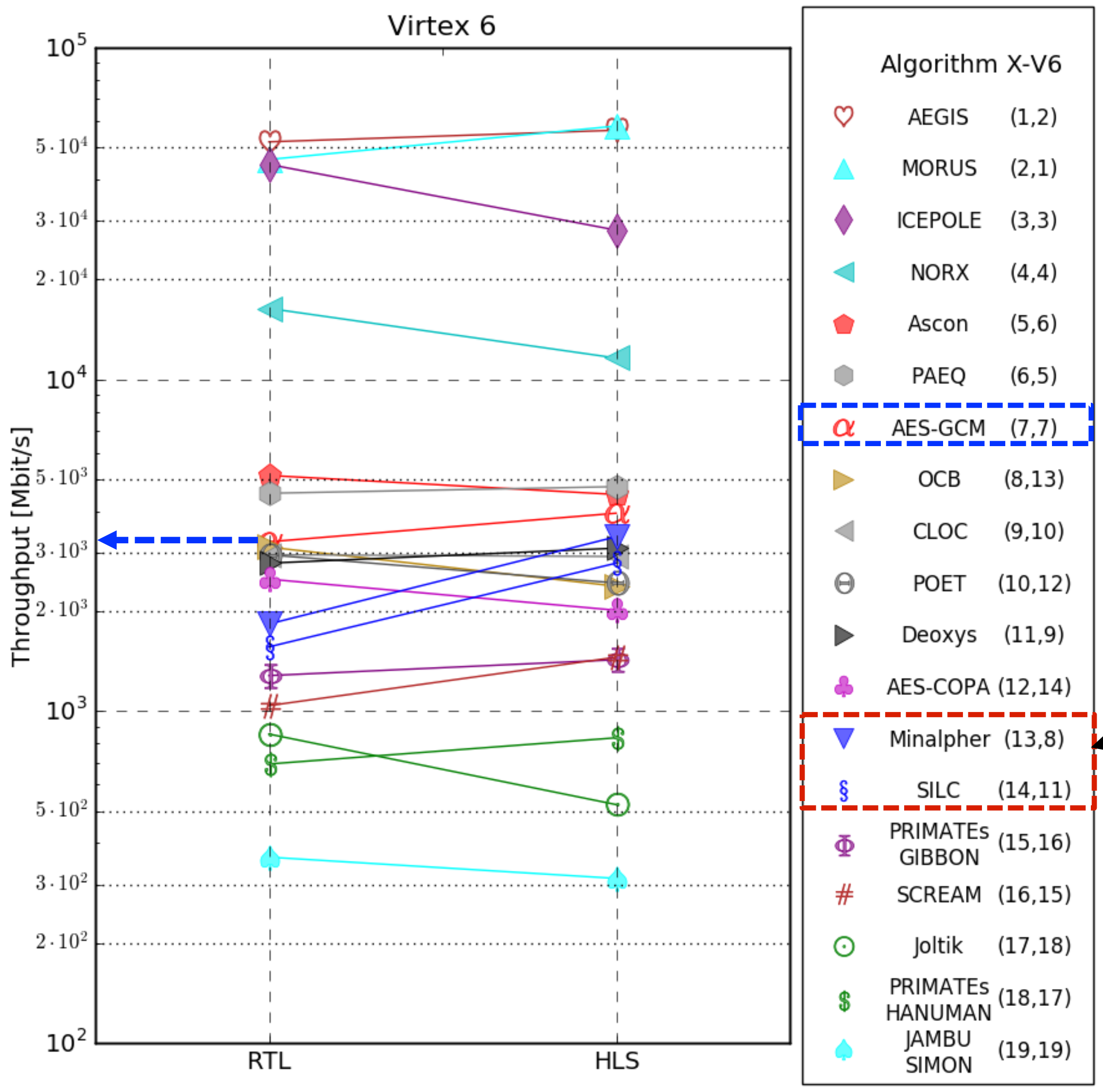
Verification Framework



Tentative Results

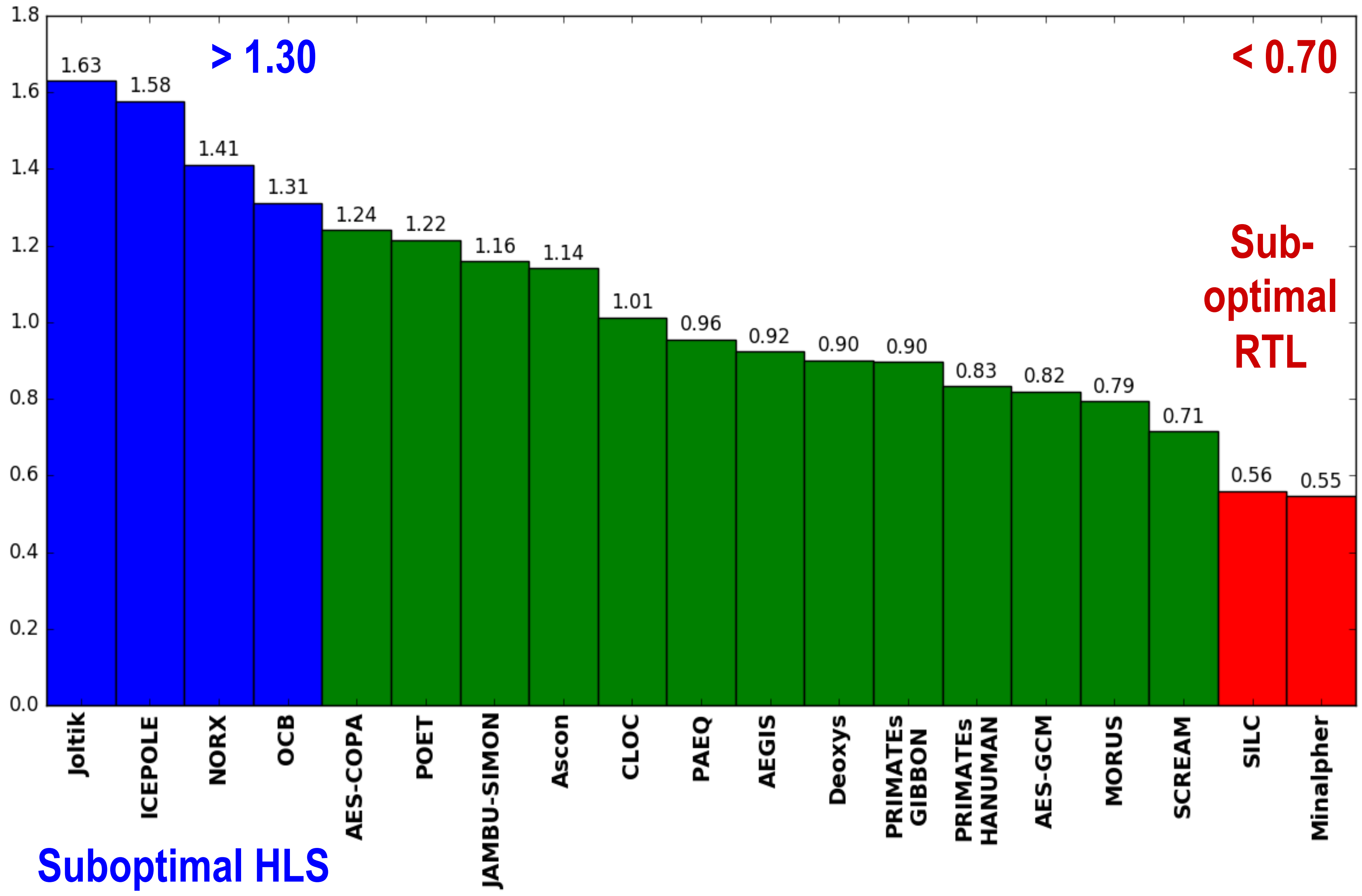
Post-Round 2 RTL, First Time with CAESAR API and RTL designers from multiple groups

RTL vs. HLS Throughput [Mbits/s]



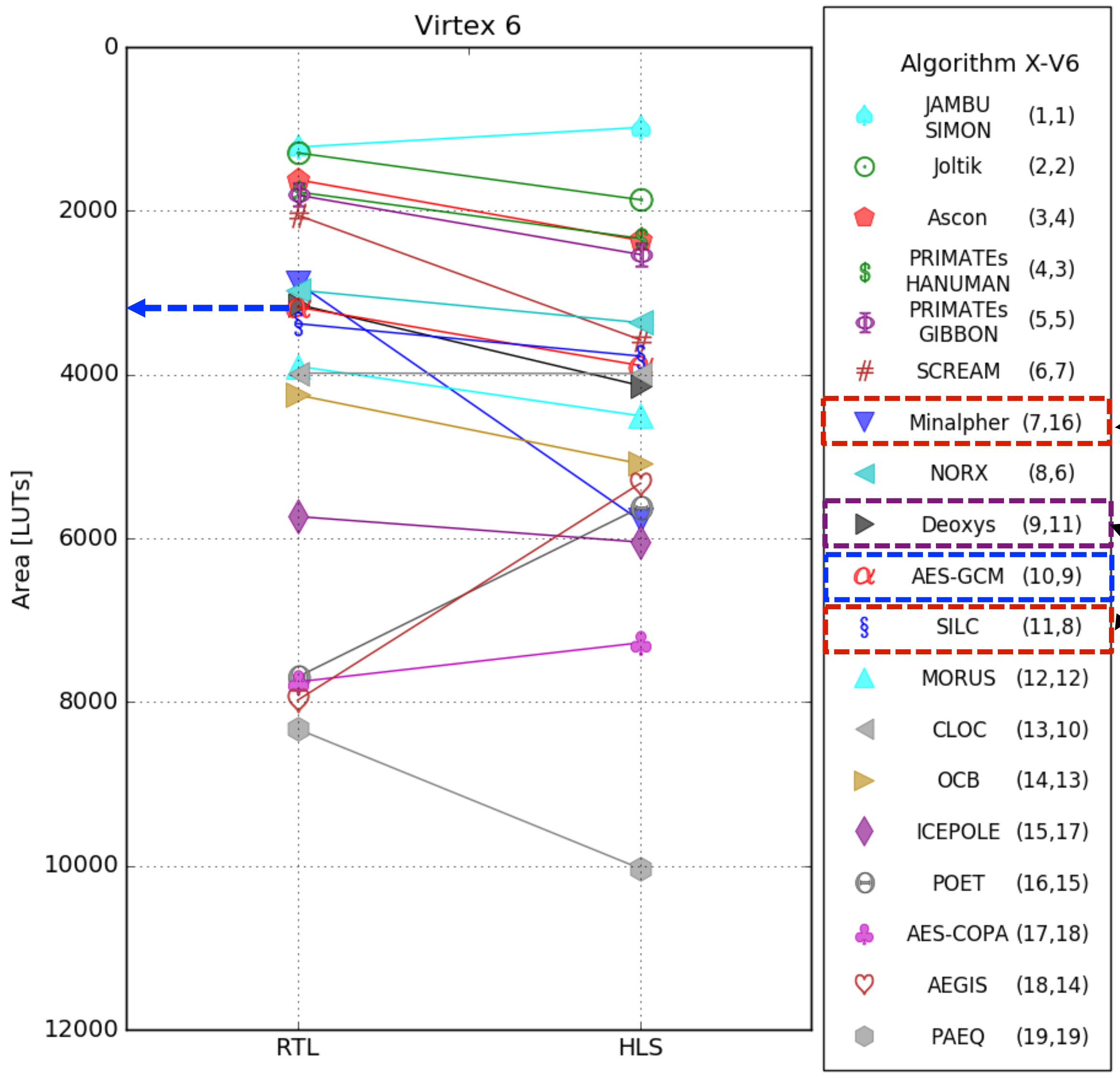
Different hardware architectures in HLS vs. RTL

RTL vs. HLS Ratios for Throughput in Virtex 6



Suboptimal HLS

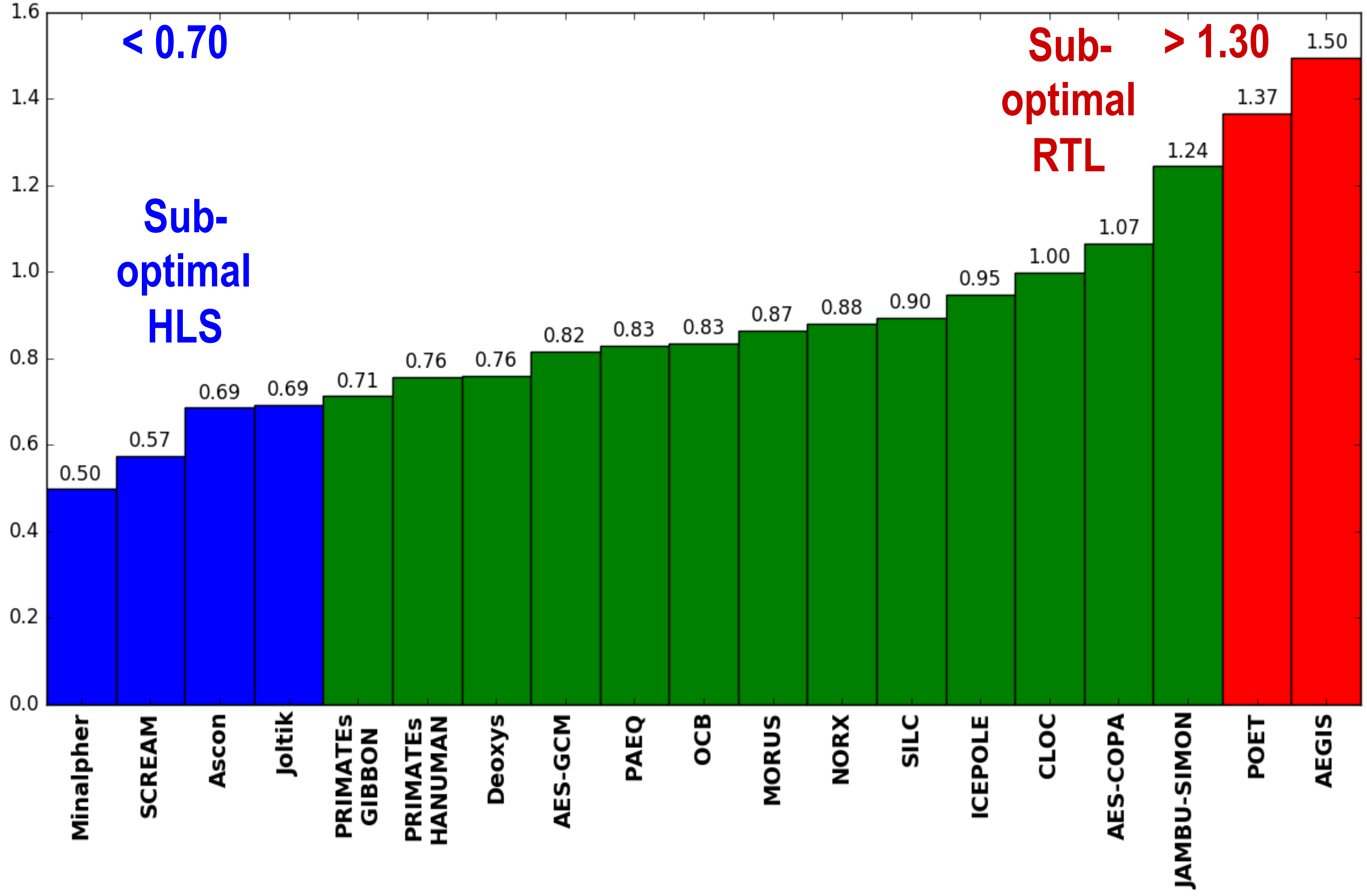
RTL vs. HLS Area [LUTs]



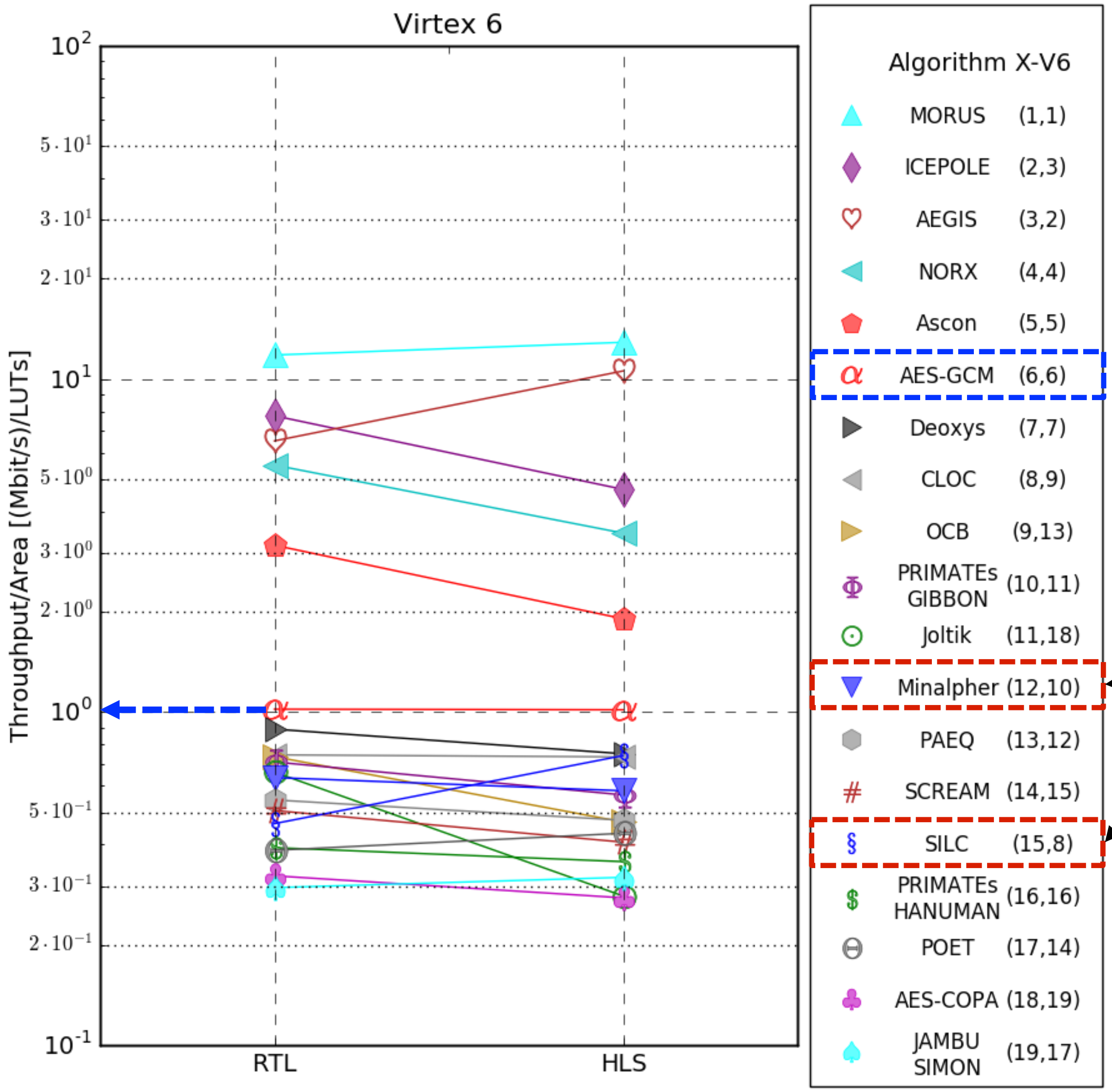
Different hardware architectures in HLS vs. RTL

Small difference in RTL

RTL vs. HLS Ratios for Area in Virtex 6



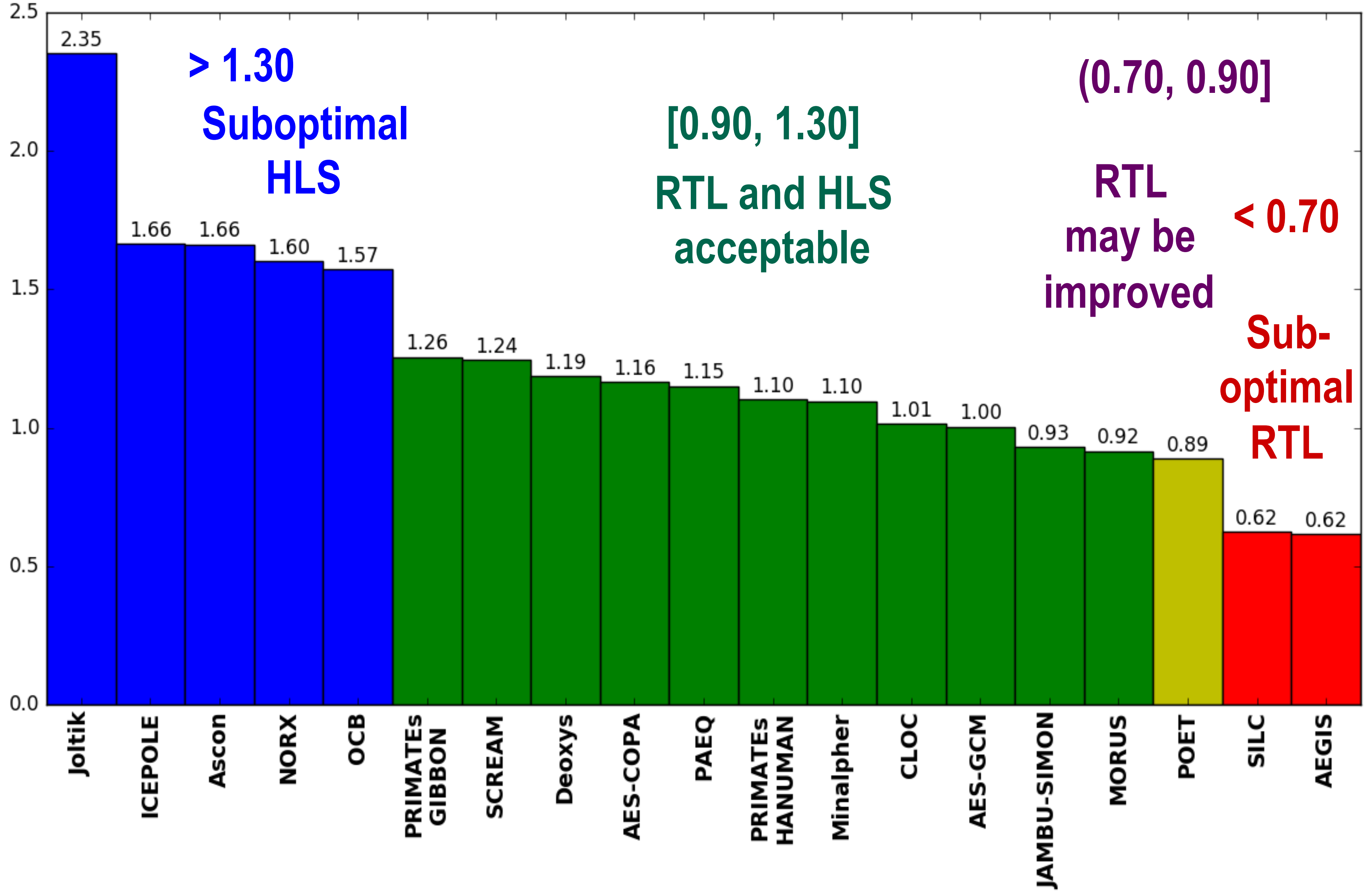
RTL vs. HLS Throughput/Area [(Mbits/s)/LUTs]



Algorithm	X-V6
▲ MORUS	(1,1)
◆ ICEPOLE	(2,3)
♡ AEGIS	(3,2)
◀ NORX	(4,4)
⬟ Ascon	(5,5)
α AES-GCM	(6,6)
▶ Deoxys	(7,7)
◀ CLOC	(8,9)
▶ OCB	(9,13)
Φ PRIMATEs GIBBON	(10,11)
⊙ Joltik	(11,18)
▼ Minalpher	(12,10)
⬢ PAEQ	(13,12)
# SCREAM	(14,15)
§ SILC	(15,8)
$ PRIMATEs HANUMAN	(16,16)
⊖ POET	(17,14)
♣ AES-COPA	(18,19)
♠ JAMBU SIMON	(19,17)

Different hardware architectures in HLS vs. RTL

RTL vs. HLS Ratios for Throughput/Area in Virtex 6



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

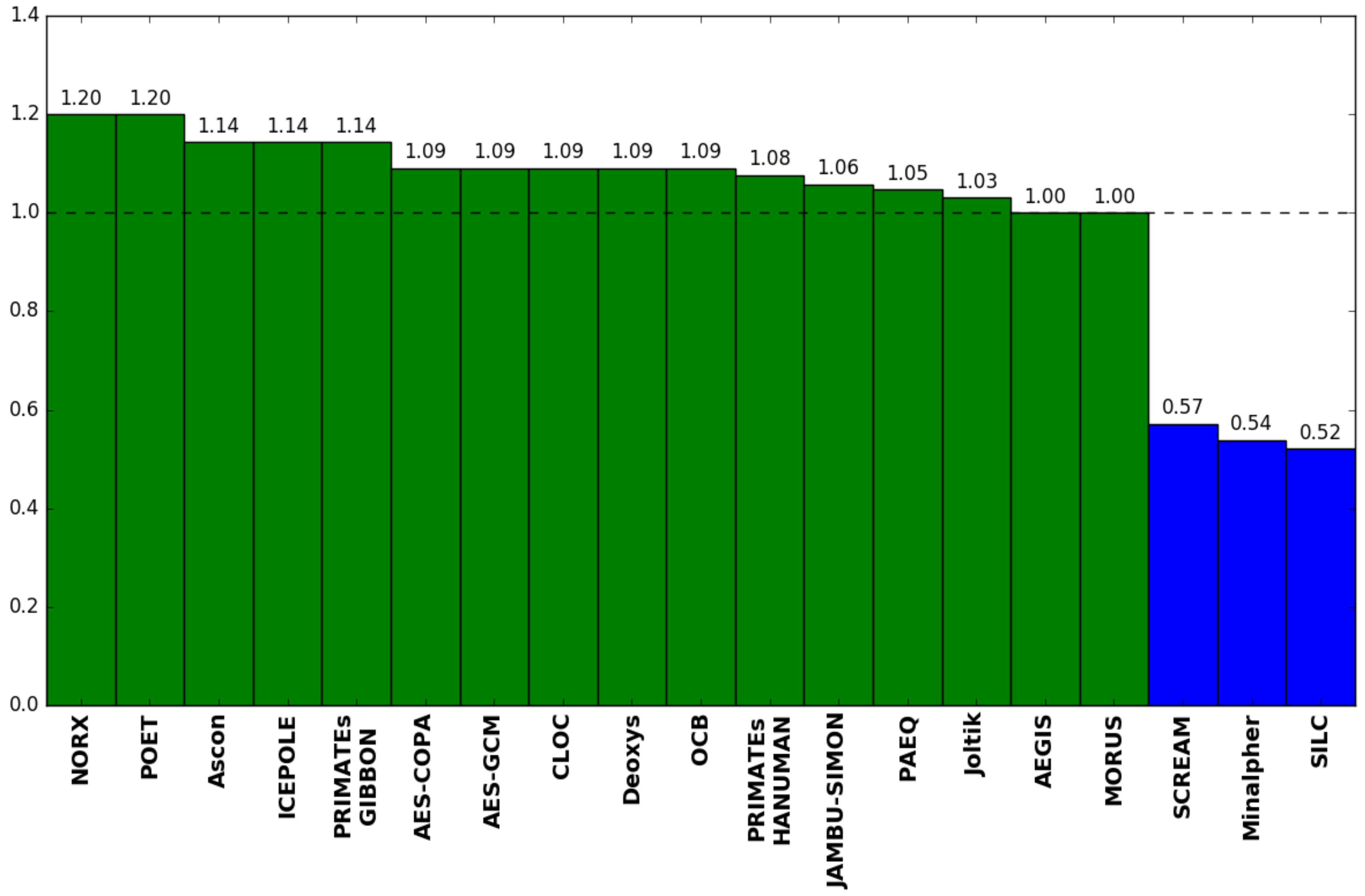
Remaining Difficulties

- **Suboptimal control unit** of HLS implementations

  #cycles per block ≥ #rounds + 2

- **Wide range of** RTL to HLS **performance metric ratios**
  Wide range of **RTL designer skills** and **selected architectures**
- A few potentially **suboptimal HLS or RTL implementations**
- **Dependence** of results **on particular FPGA family**
- Efficient and reliable **generation of HLS-ready C/C++** code
- **Portability among** HLS **tools**
- **Licensing limitations of commercial tools**

HLS vs. RTL Ratios for Number of Clock Cycles



Best HLS/RTL reported so far

- "A Survey and Evaluation of FPGA High-Level Synthesis Tools"
- IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume: 35, Issue: 10, Oct. 2016)
  - 12 leading researchers in the HLS field
  - Co-developers of top 3 academic HLS Tools

Tools	aes-encrypt	aes-decrypt	sha	blowfish
Best/Manual	<u>60</u>	129	<u>2.5</u>	3.1

Typical Doubts (from reviewers of our papers)

- How can we trust these tools?

  If HLS used efficiently, maximum 20% penalty
  in the number of clock cycles per block.
  Easy to verify by comparing vs. the number of rounds.

- Isn't manual design always better?

  Multiple HLS designs with one or more metrics better.
  7 out of 19 HLS designs with better Throughput/Area.

- Is it fair to compare manual designs with HLS designs?

  It is not our intention. HLS results are supposed
  to be compared with HLS only. However if an
  existing RTL result worse, it is OK to use HLS result
  temporarily.

Ekawat Homsirikamol
a.k.a "Ice"

- Main developer of the **RTL Round 2 Benchmarking Framework** and **Developer's Package**
- **RTL Designer for 12 Round 2 Candidates: AES-GCM, AEZ, Ascon, Deoxys, HS1-SIV, ICEPOLE, Joltik, NORX, OCB, PAEQ, Pi-Cipher, STRIBOB**

- **Developer of the HLS-based methodology and framework for crypto applications**

Thank you!

Comments?

Questions?

Suggestions?

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

CERG: http://cryptography.gmu.edu