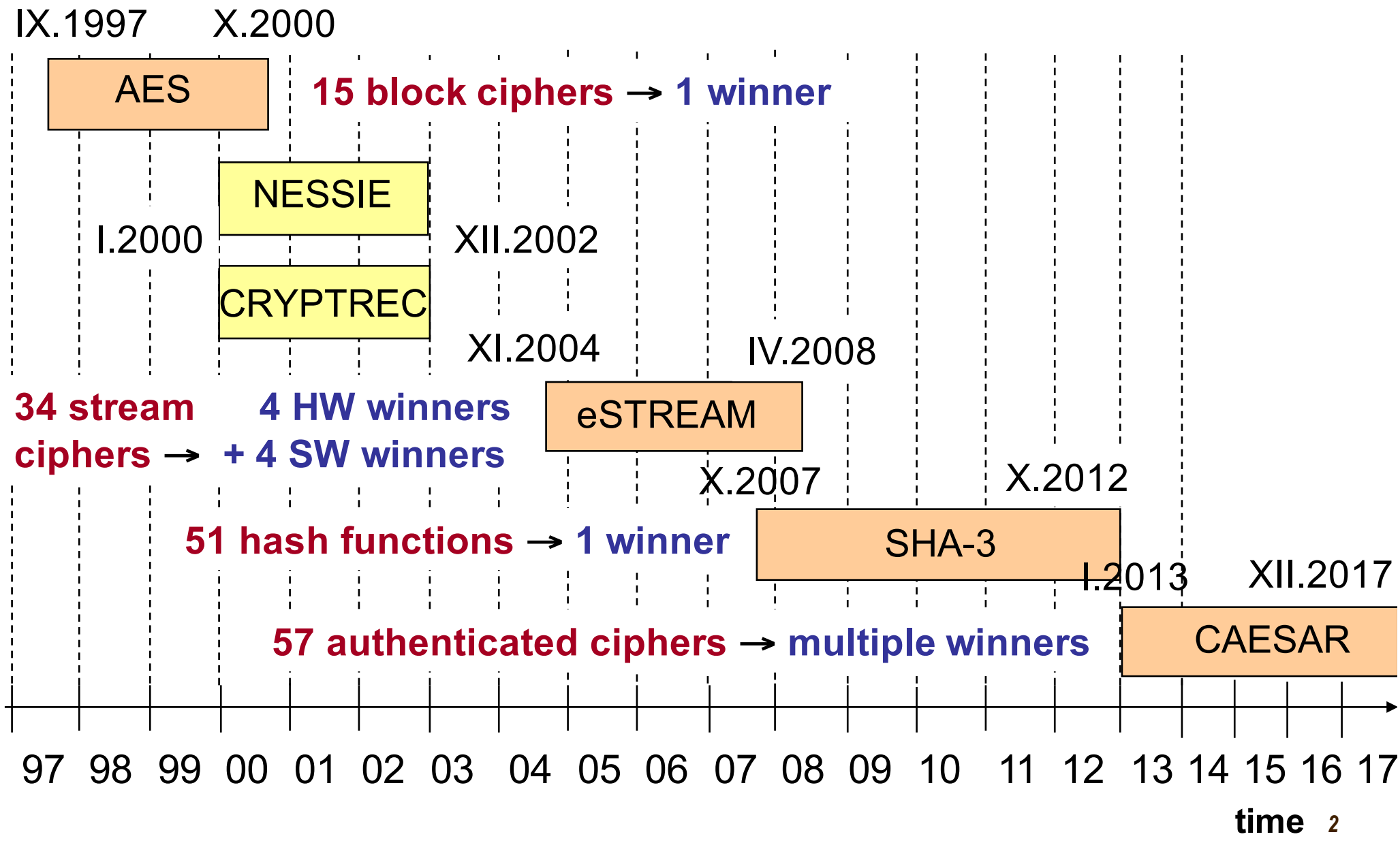
Vivado HLS Implementation of Round-2 SHA-3 Candidates

Farnoud Farahmand
ECE 646

CERG: http://cryptography.gmu.edu

Cryptographic Standard Contests



Evaluation Criteria

Security

Software Efficiency

µProcessors µControllers

Hardware Efficiency

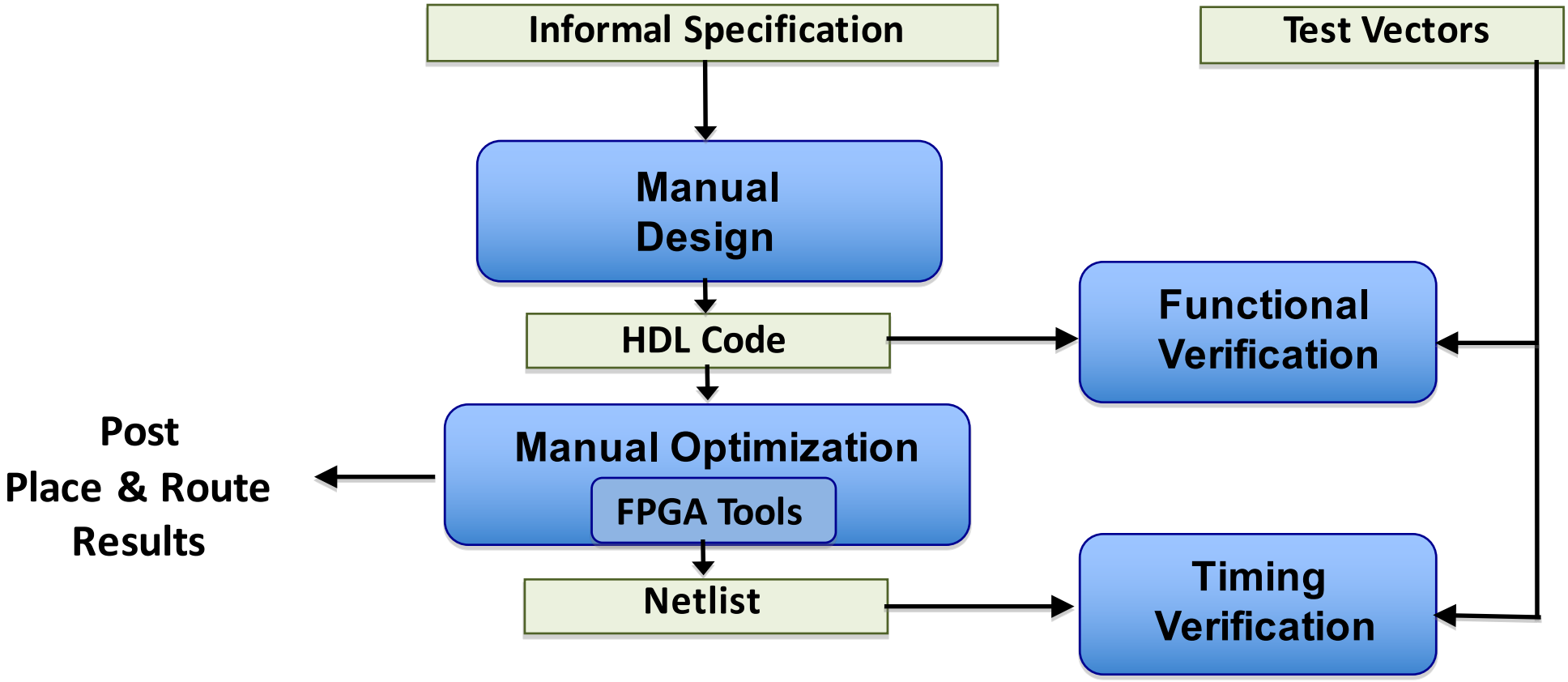
FPGAs ASICs

Flexibility

Simplicity

Licensing

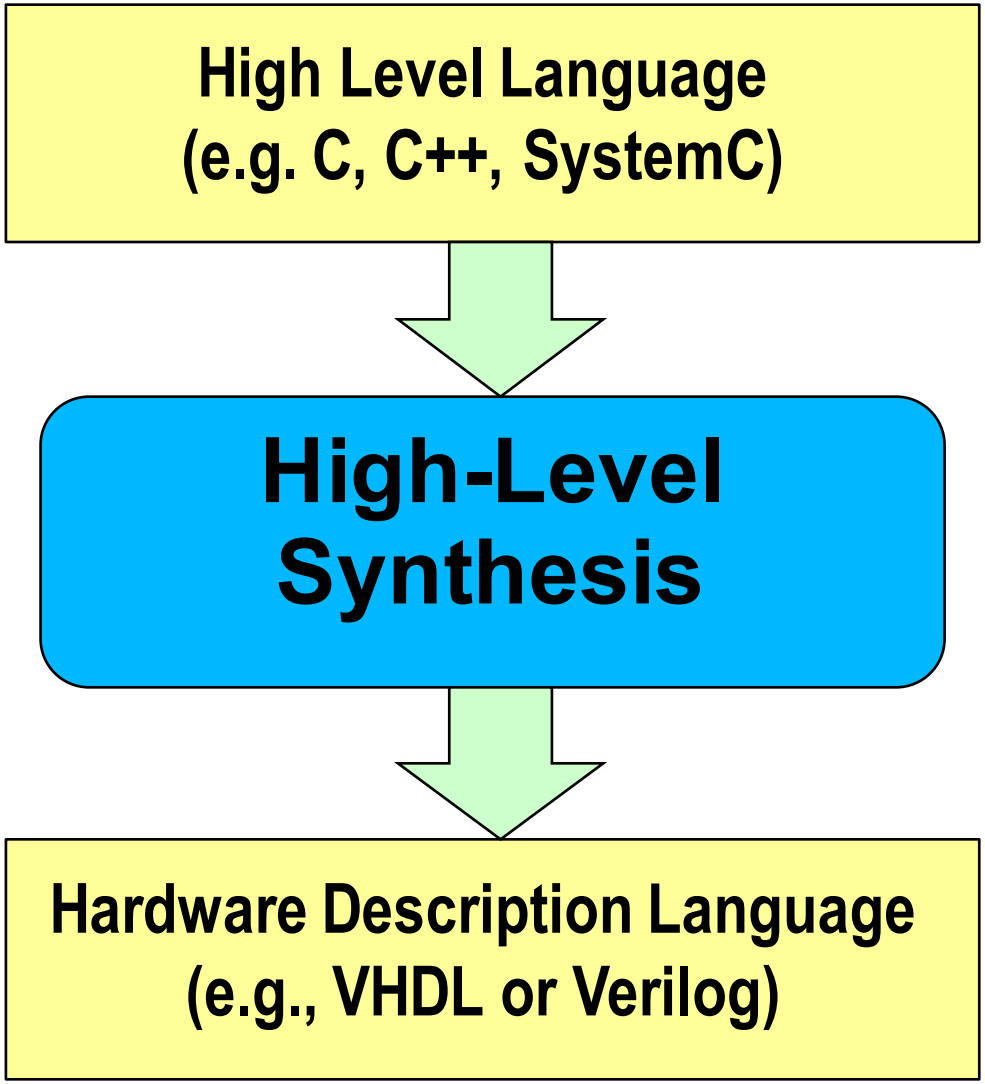
Traditional Development & Benchmarking Flow



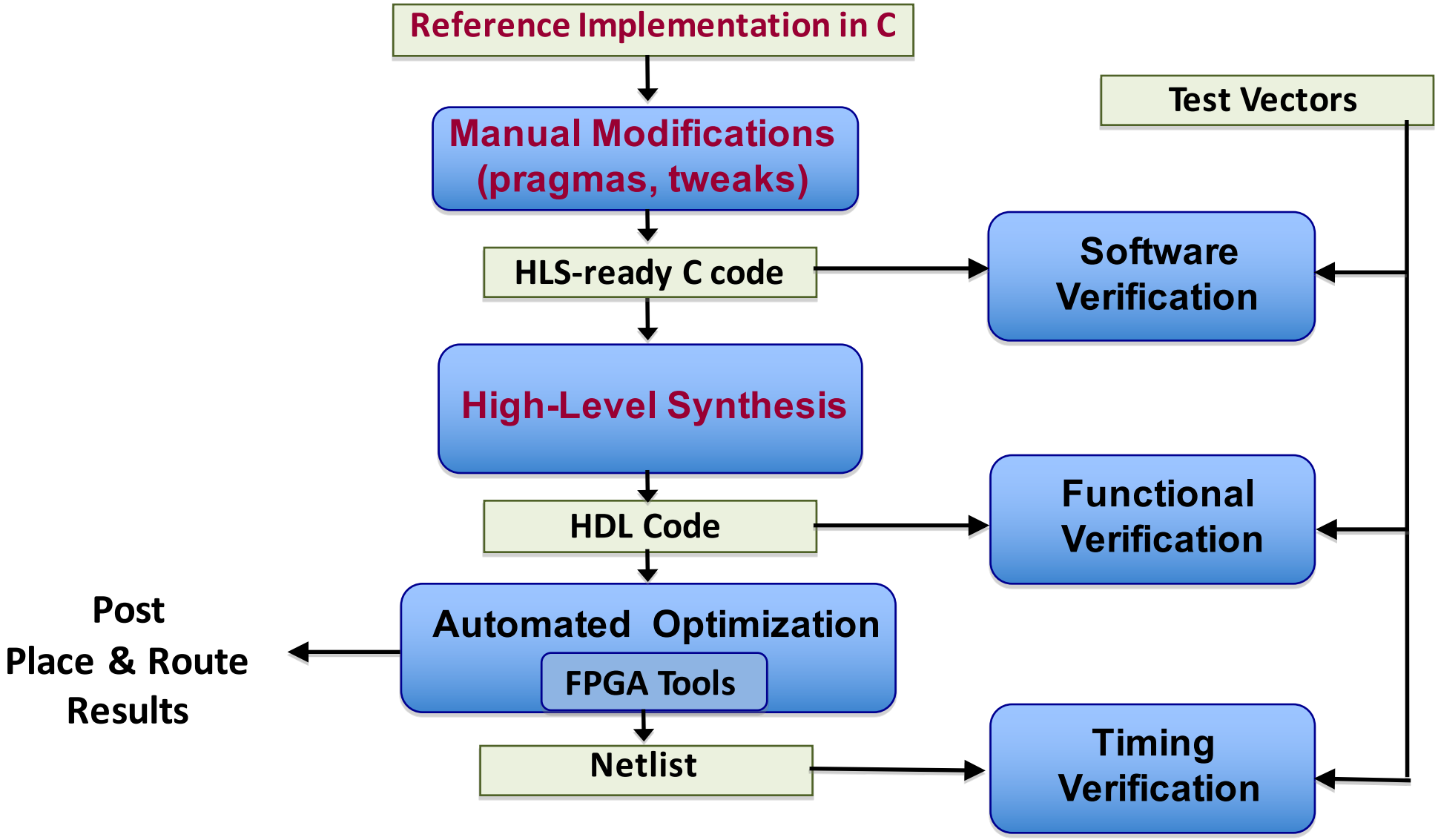
Remaining Difficulties of Hardware Benchmarking

- 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 hash value size)
- Multiple hardware architectures
- Dependence on skills of designers

High-Level Synthesis (HLS)



Proposed HLS-Based Development and Benchmarking Flow



Examples of Source Code Modifications (1)

Unrolling of loops:

```
for (i = 0; i < 16; i++)
#pragma HLS UNROLL
        b[i] = a[i]+4;
```

Array Reshaping:

```
void main()
{
 int A[16];
 #pragma HLS ARRAY_RESHAPE Variable=A complete dim=1
                ...
}
```

Examples of Source Code Modifications (2)

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);
  }
```

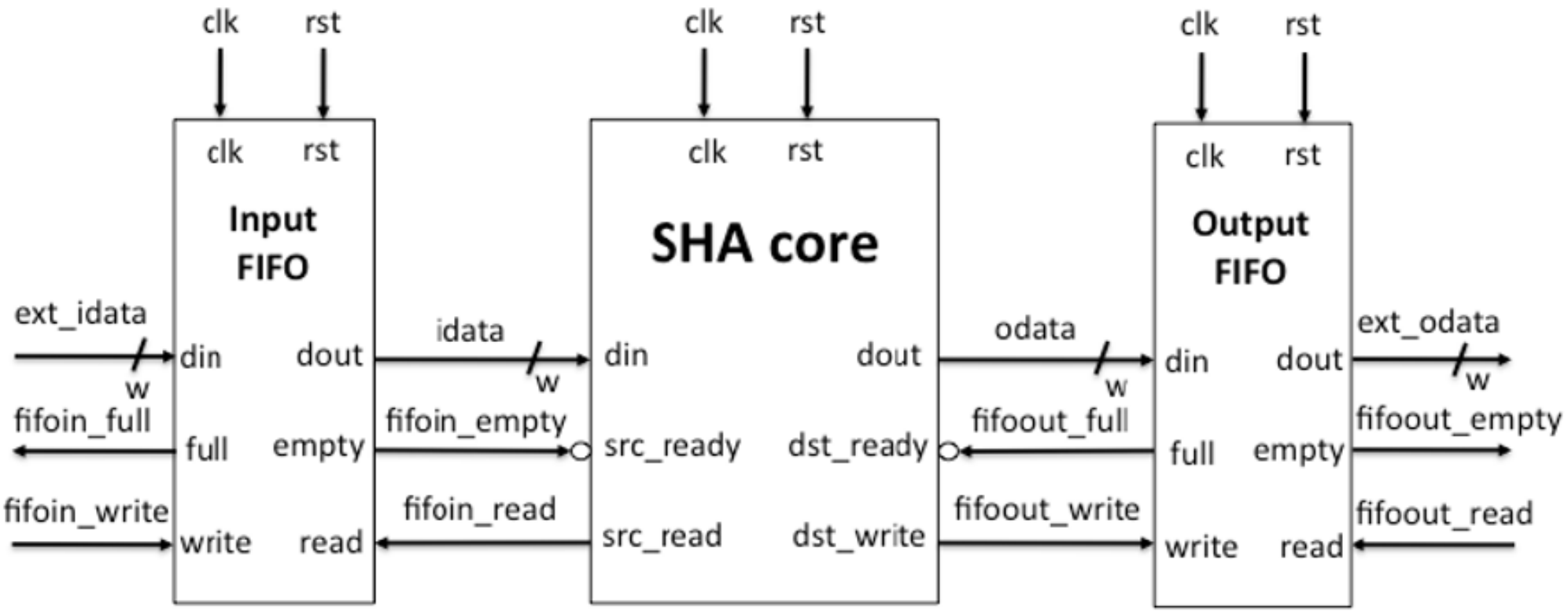
Our Test Case

- 4 Round 2 SHA-3 candidates + 5 Finalists are done previously
- Basic iterative architecture
- GMU Hardware Interface for SHA core
- Padding done in software
- RTL Implementations developed previously
- Starting point: Informal specifications and reference software implementations in C provided by the algorithm authors
- Post P&R results generated for
  - Xilinx Virtex 5 using Xilinx ISE
- No use of BRAMs or DSP Units

Parameters of Hash Functions

Algorithm	Number of Rounds	Block Size	Hash size	Chain Size
Luffa	8	256	256	768
Cubehash	16	256	256	1024
BMW	1	512	256	512
ECHO	8	1536	256	512

SHA core Interface

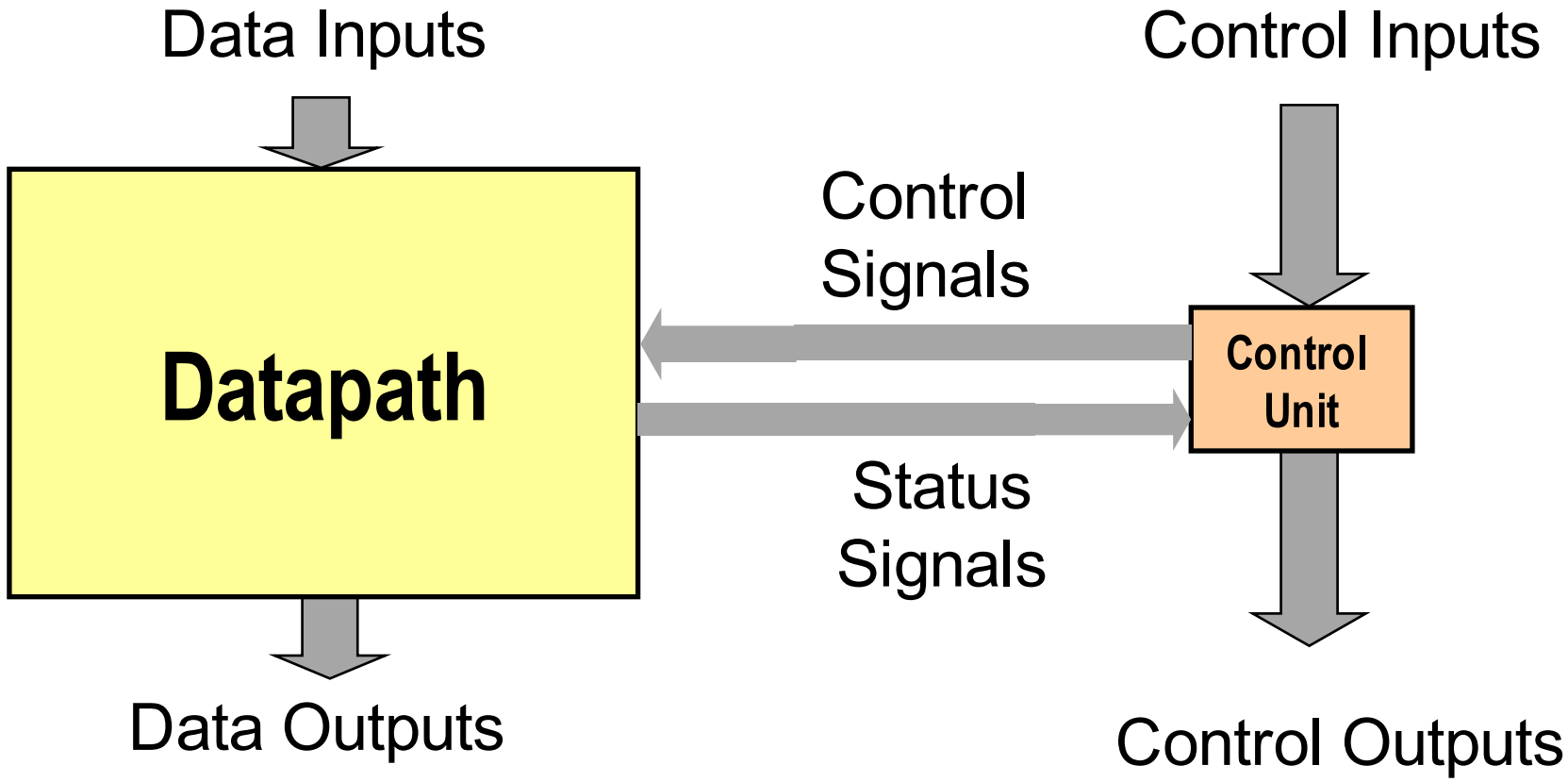


HLS and RTL Implementations Parameters

Algorithm	#Rounds	Cycles/Block RTL	Cycles/Block HLS	Interval HLS
Luffa	8	9	10	11
Cubehash	16	16	17	18
BMW	1	2	1	1
ECHO	8	26	25	26

Interval = Number of Cycles require to process two consecutives blocks

Datapath vs. Control Unit



Determines

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

Determines

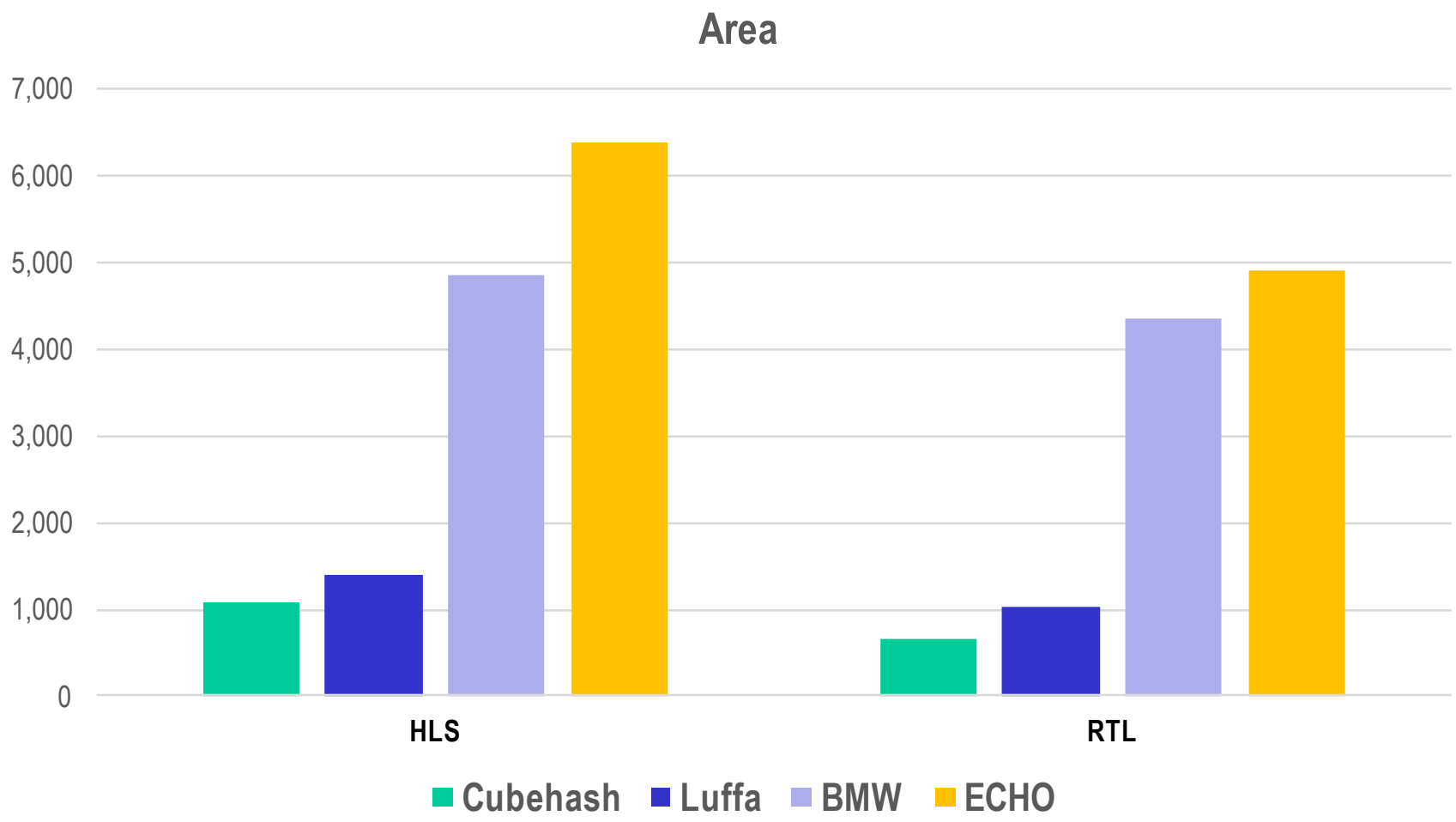
- **Number of clock cycles**

Encountered Problems

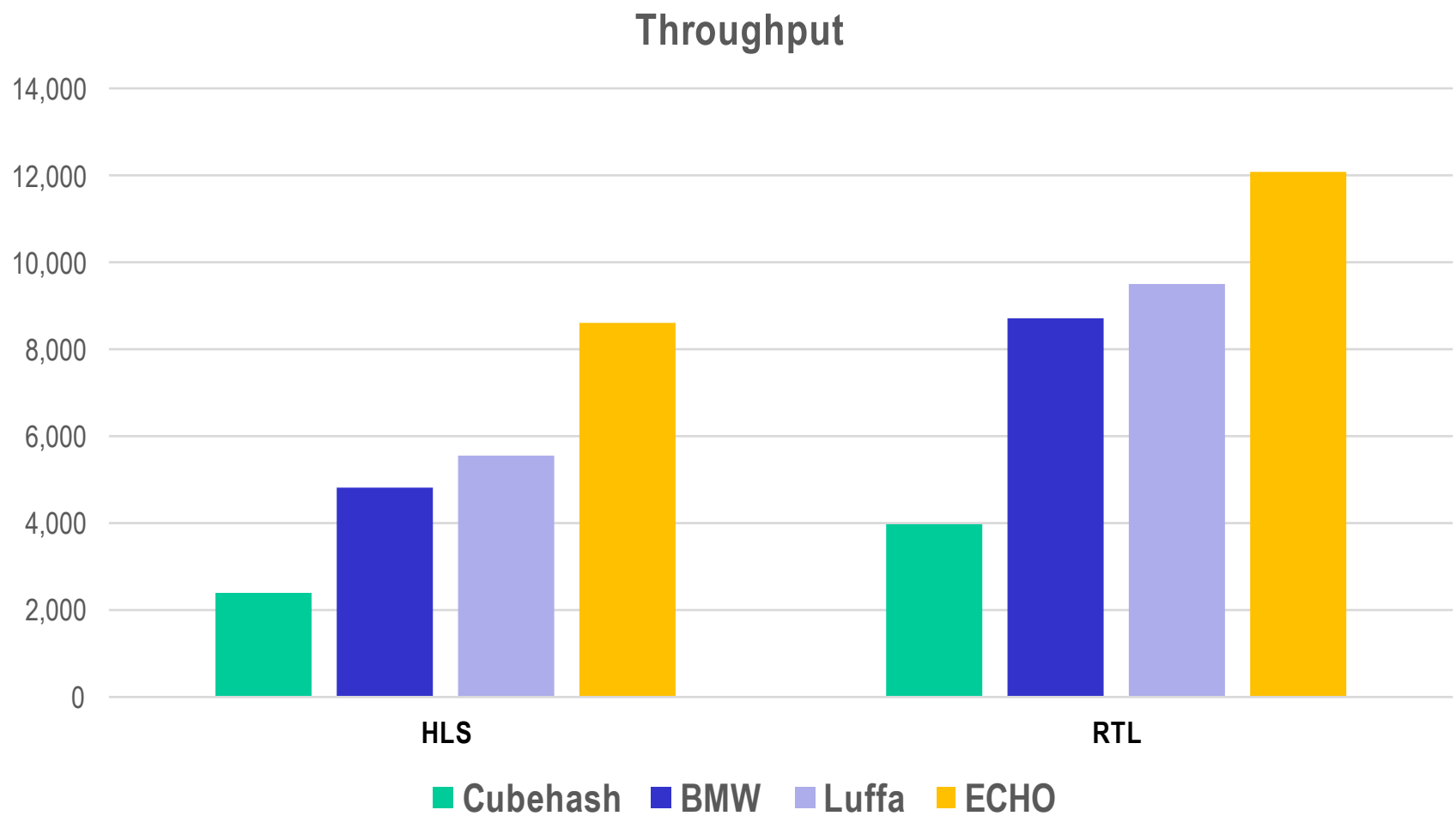
Control Unit suboptimal

- 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

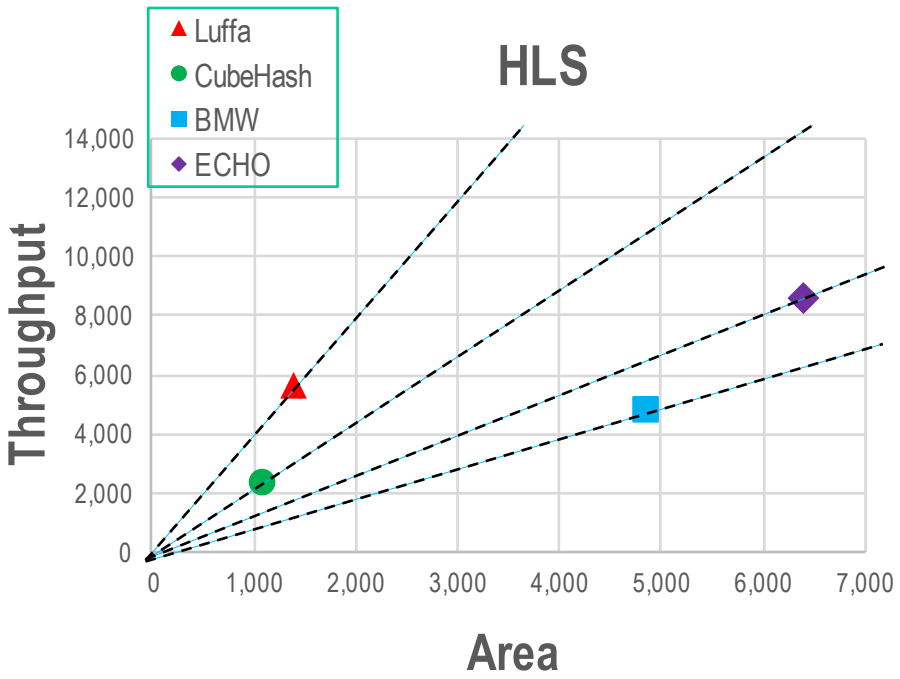
RTL vs. HLS (Area):

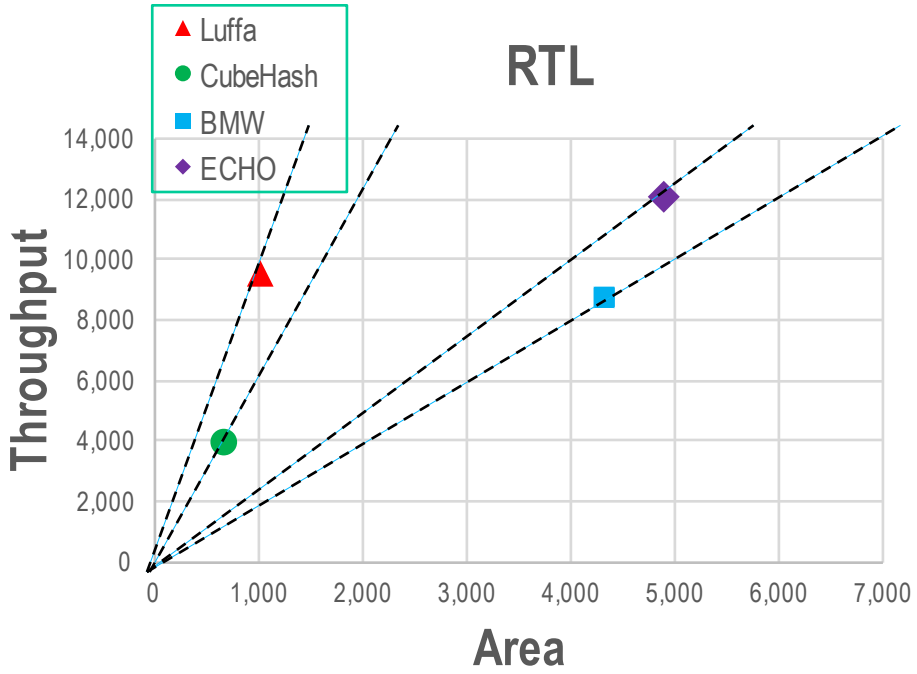


RTL vs. HLS (Throughput):



Throughput vs. Area:





Overall Results:

	HLS				RTL			
	Freq.	TP	A	TP/A	Freq	TP	A	TP/A
Luffa	217.39	5,565	1,390	4.00	334.4	9,513	1,023	9.29
Cubehash	168.71	2,399	1,076	2.23	248.3	3,972	663	5.99
BMW	9.45	4,838	4,859	0.99	34	8,704	4,350	2.00
ECHO	145.77	8,612	6,389	1.34	203.8	12,094	4,888	2.47

	RTL/HLS			
	Freq.	TP	A	TP/A
Luffa	1.53	1.70	0.73	2.32
Cube Hash	1.47	1.65	0.61	2.68
BMW	3.59	1.79	0.89	2.00
ECHO	1.39	1.40	0.76	1.83

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 4 Round 2 SHA-3 candidates demonstrated correct ranking for all 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?

CERG: http://cryptography.gmu.edu