FPGA Benchmarking for High-Speed and Medium-Speed Implementations

Kris Gaj
George Mason University
U.S.A.

Codes developed and results generated by:

Ekawat Homsirikamol

Marcin Rogawski

Malik Umar Sharif

Rabia Shahid

Bilal Habib

Outline

- Flexibility of the SHA-3 candidates in hardware in terms of speed-area trade-offs using
  - a. folding
  - b. unrolling
  - c. pipelining
  - d. embedded FPGA resources (embedded DSP units, memory blocks, etc.)
- FPGA performance metrics
- Summary of GMU results
- Presentation and comparison of FPGA results from various groups

  Discussion of possible sources of discrepancies
- Pros and Cons of all candidates

High-Speed Architecture of a Hash Function

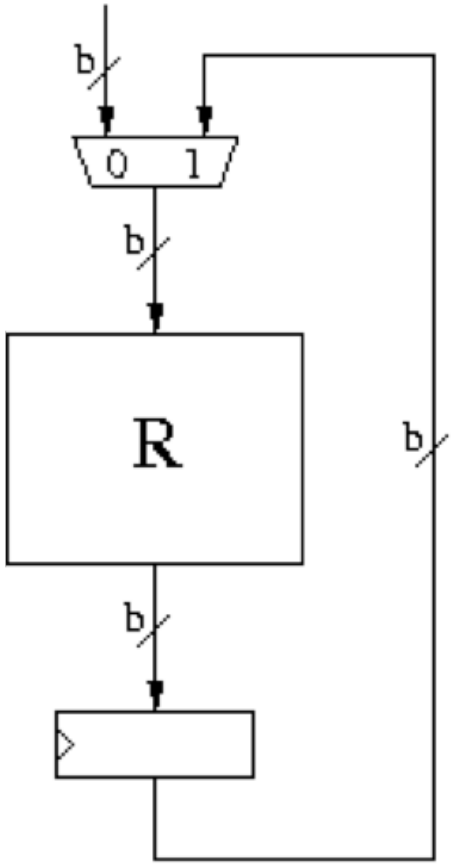
Features

- datapath width = state size
- optimization for throughput or throughput to area ratio
  [rather than for area or power]
- typically [but not always] one clock cycle per one round/step

Our Target

High-Speed Architecture Optimized for
the Maximum Throughput to Area Ratio

Starting Point: Basic Iterative Architecture



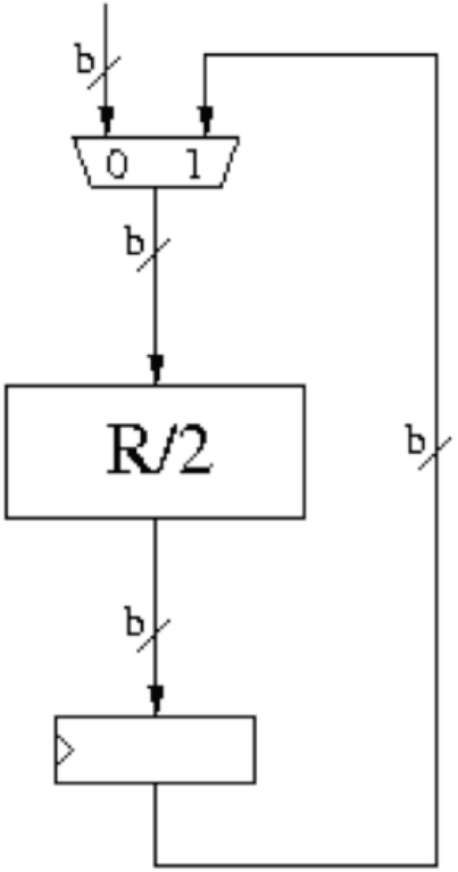
- datapath width = state size
- one clock cycle per one round/step

Block processing time = #R · T

#R = number of rounds/steps
T = clock period

Currently, most common architecture used to implement SHA-1, SHA-2, and many other hash functions.

Horizontal Folding



- datapath width = state size
- two clock cycles per one round/step

Block processing time = (2·#R) * T'

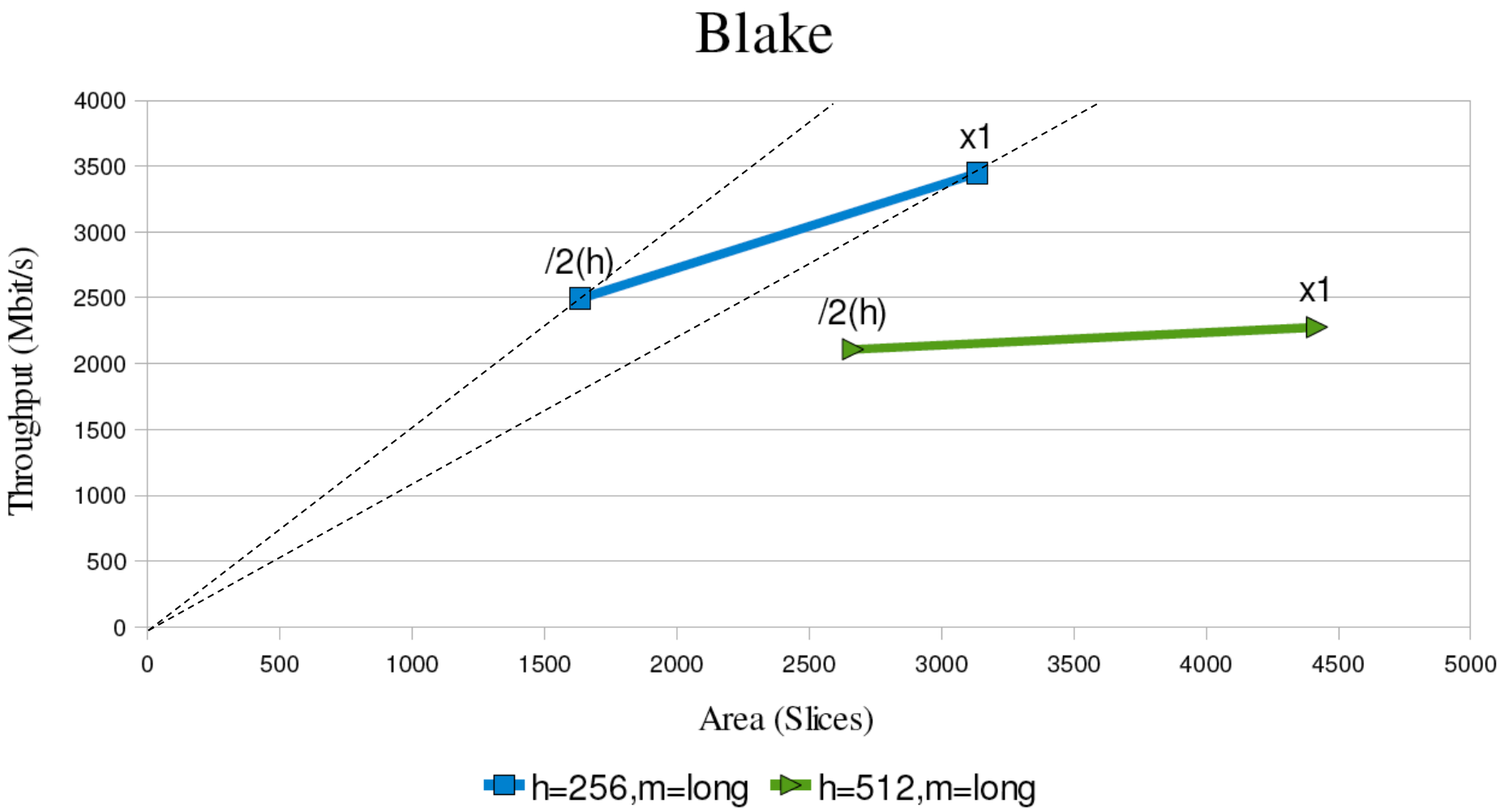
$T/2 < T' < T$

typically $T' \approx T/2$

$Area/2 < Area' < Area$

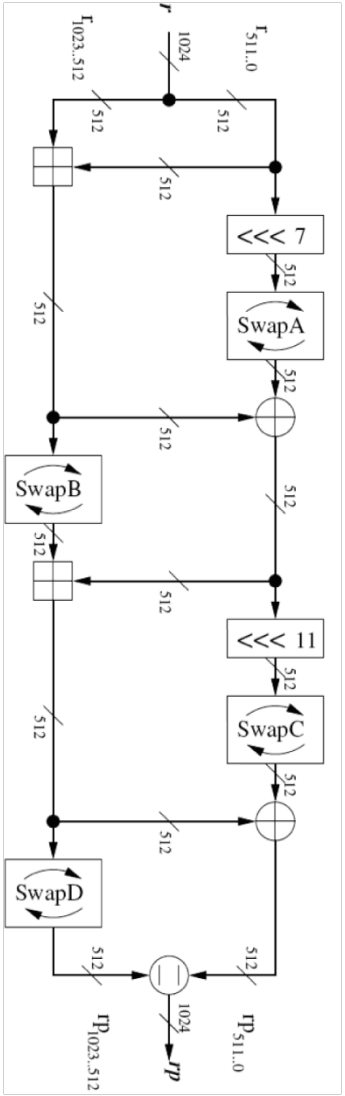
Typically Throughput/Area ratio increases

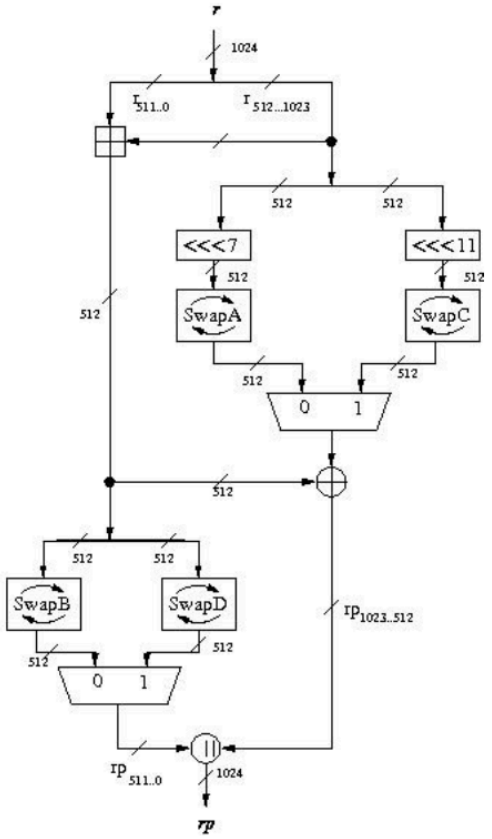
Horizontally Folded vs. Basic Iterative BLAKE



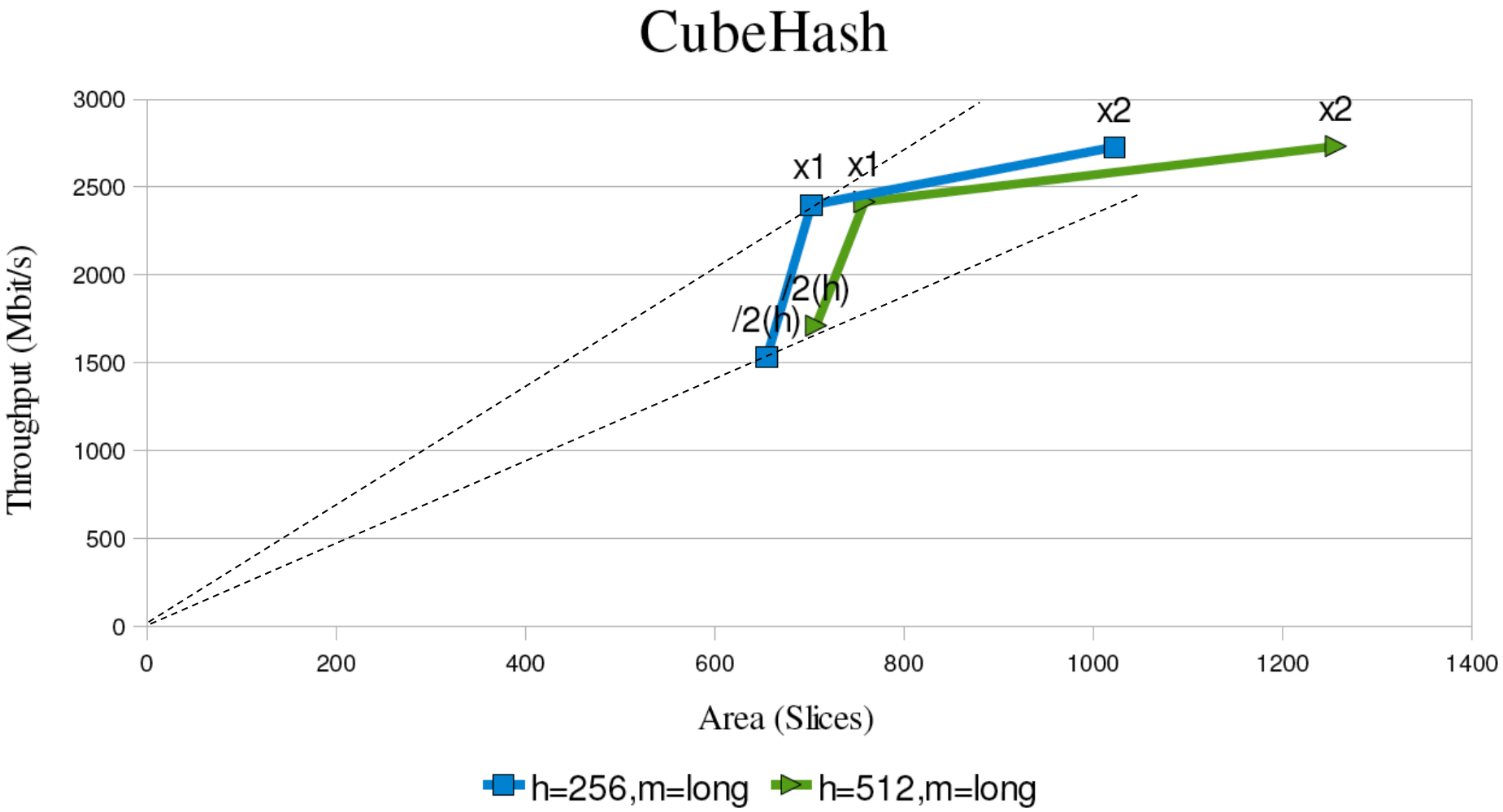
x1 – basic iterative /2(h) – folded horizontally by a factor of 2

Horizontal folding of CubeHash





Horizontally Folded vs. Basic Iterative CubeHash



x1 – basic iterative /2(h) – folded horizontally by a factor of 2

x2 – two times unrolled

SHA-3 Candidates Benefiting from Horizontal Folding

BLAKE: two layers of G-functions - folding by 2 : /2

Fugue-256: two iterations of (ROR3, CMIX, SMIX) - folding by 2 : /2

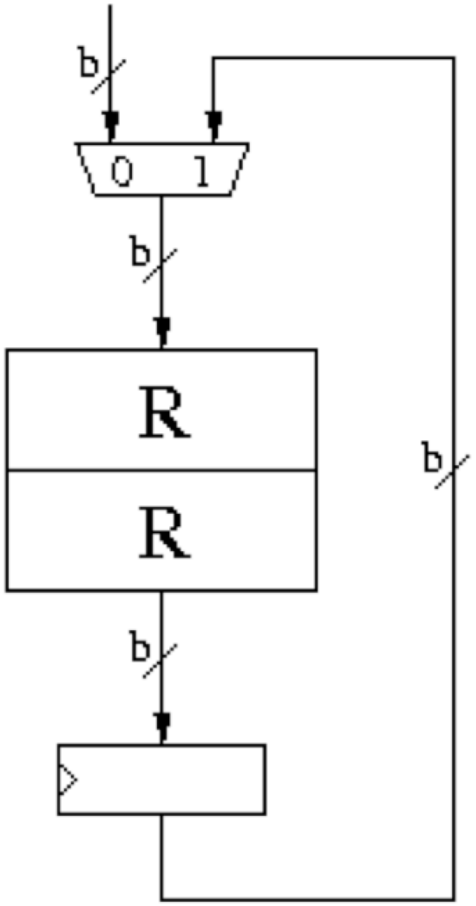
Fugue-512: four iterations of (ROR3, CMIX, SMIX) - folding by 4: /4

ECHO: two layers of BIG.SubBytes - folding by 3/2: x2/3

SHAvite-3-256: three iterations of AES Round - folding by 3 : /3

SHAvite-3-512: four iterations of AES Round - folding by 4: /4

Unrolling



- datapath width = state size
- one clock cycle per two rounds

Block processing time = (#R/2) * T'

$T < T' < 2 \cdot T$

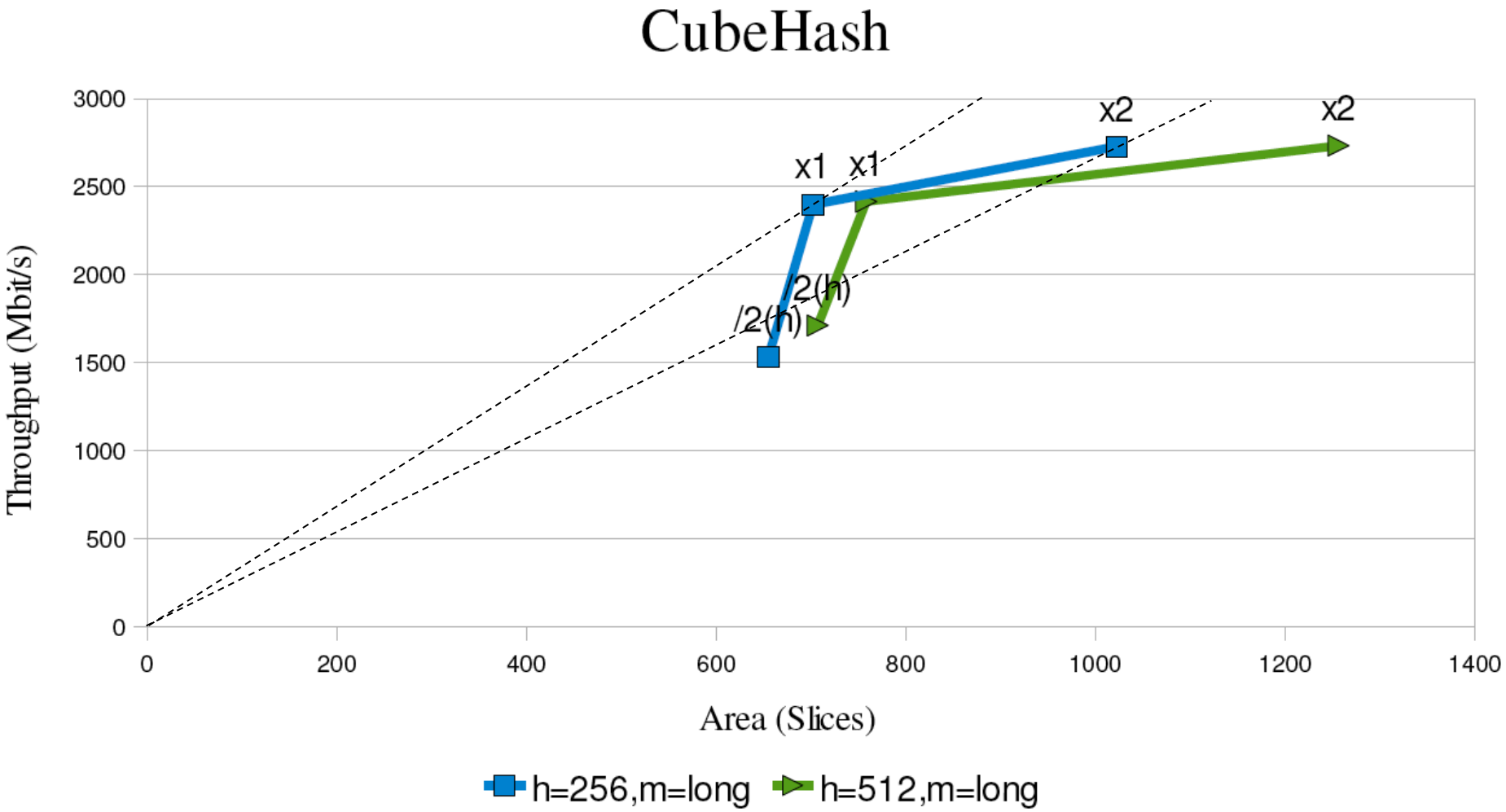
typically $T' \approx 2 \cdot T$

$Area/2 < Area' < 2 \cdot Area$

Typically $Area' \approx 2 \cdot Area$

Typically Throughput/Area ratio decreases

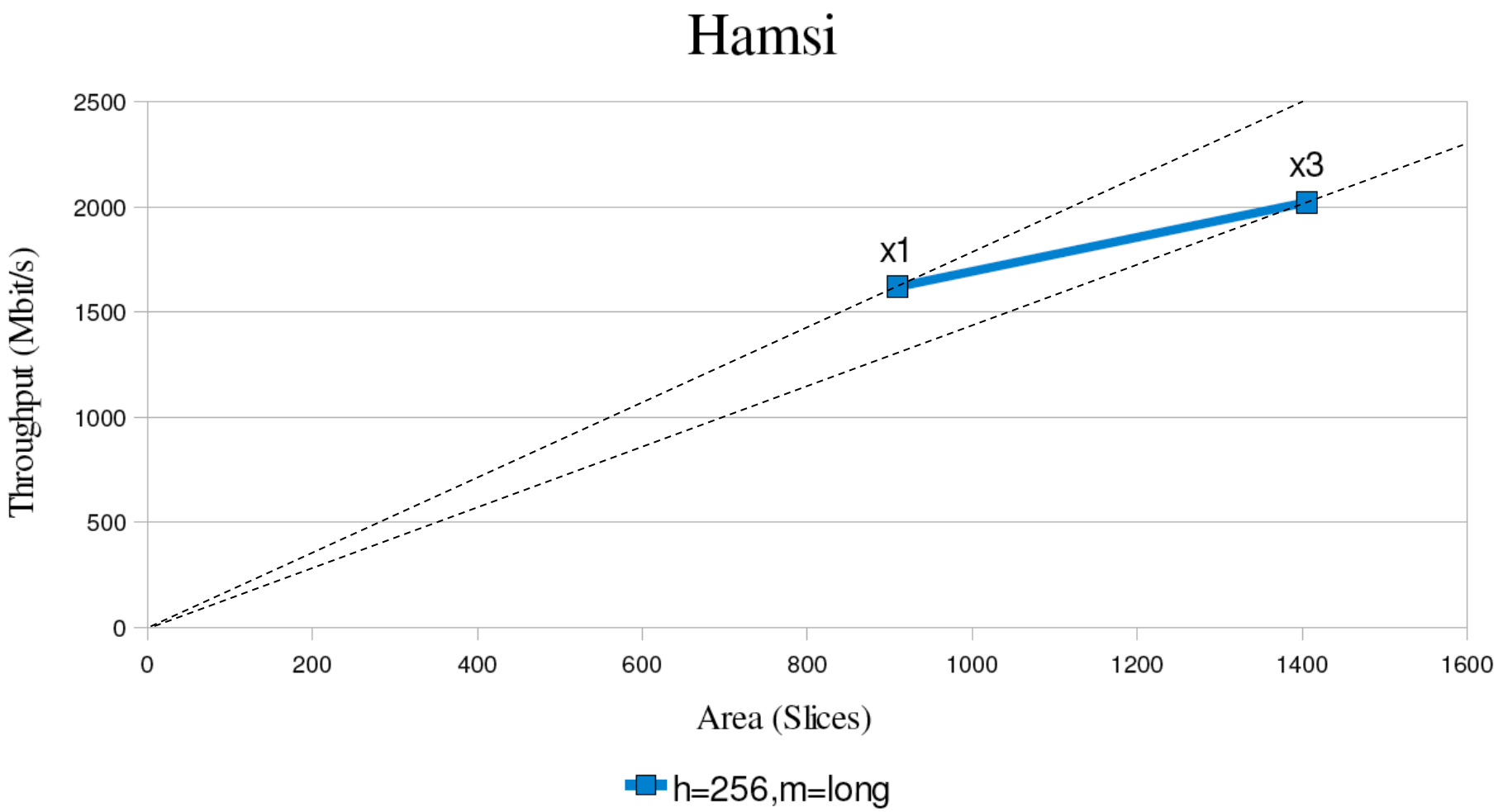
Unrolling of CubeHash



x1 – basic iterative /2(h) – folded horizontally by a factor of 2

x2 – two times unrolled

Unrolling of Hamsi



x1 – basic iterative x3 – unrolled by a factor of 3

How Can Functions Benefit from Unrolling?

Functions having non-uniform rounds/steps can benefit from unrolling.

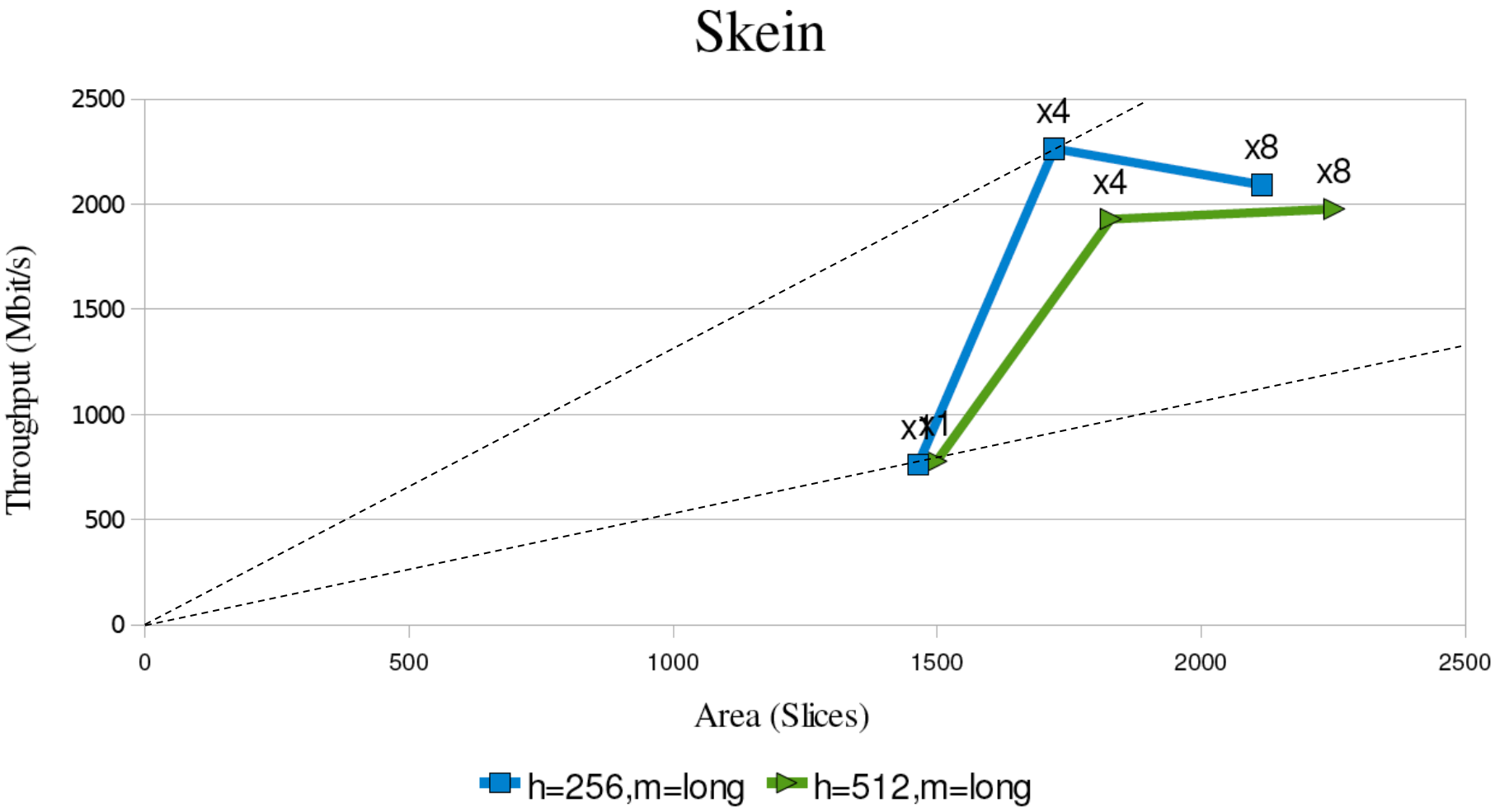
Examples:

Skein: 8 consecutive rounds use 8 different rotation amounts.

SIMD: 36 steps use 16 different rotation amounts.

The logic of a single round may be significantly simplified as a result of unrolling.

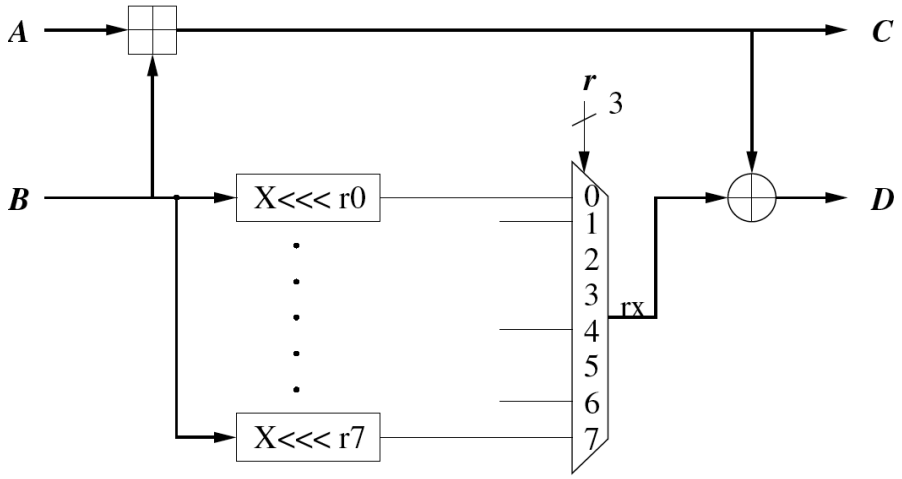
Unrolling Skein



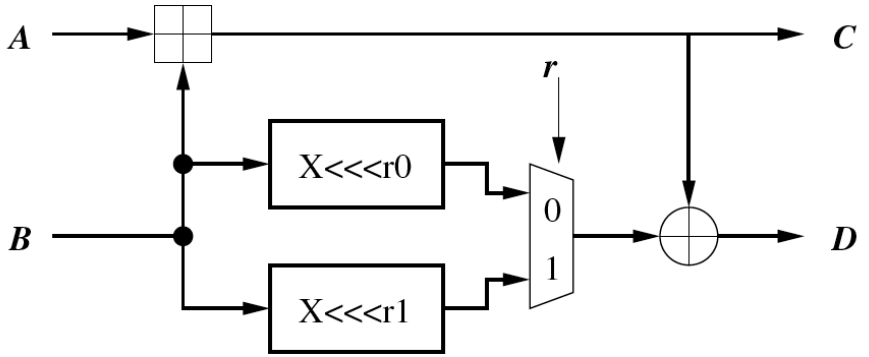
x1 – basic iterative x4 – unrolled by a factor of 4 x8 – unrolled be a factor of 8

Basic operation in Skein x1 and Skein x4

Basic operation, MIX, in Skein x1
(basic iterative)

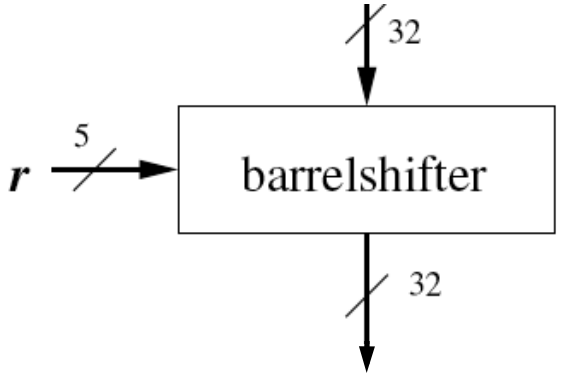


Basic operation, MIX, in Skein x4
(4 times unrolled)

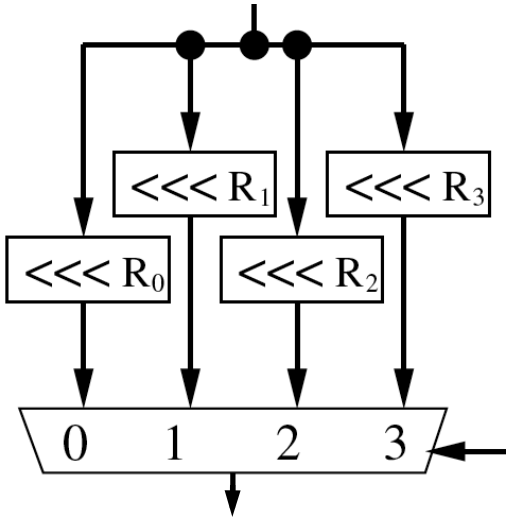


Basic operation in SIMD x1 and SIMD x4

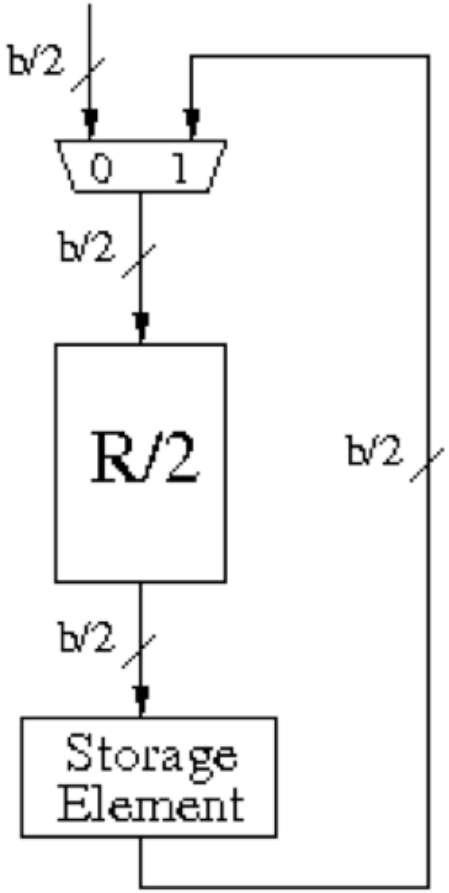
Operation required in SIMD x1 (basic iterative)



Equivalent operation in SIMD x4 (4 times unrolled)



How to Reduce Area? – The case for Vertical Folding



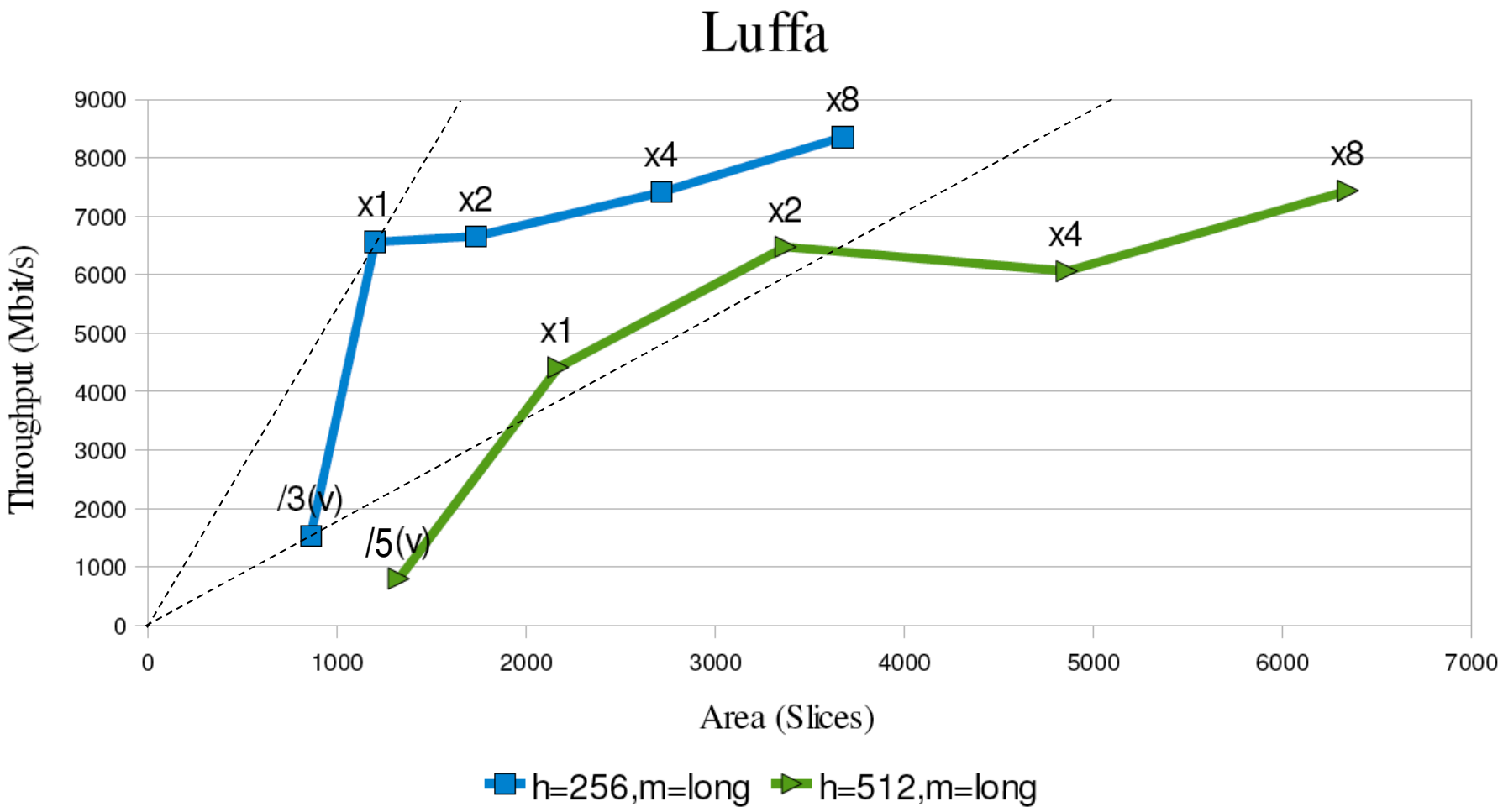
- datapath width = state size/2
- two clock cycles per one round/step

Block processing time = (2·#R) * T'

typically T' ≈ T

Area/2 < Area' < Area

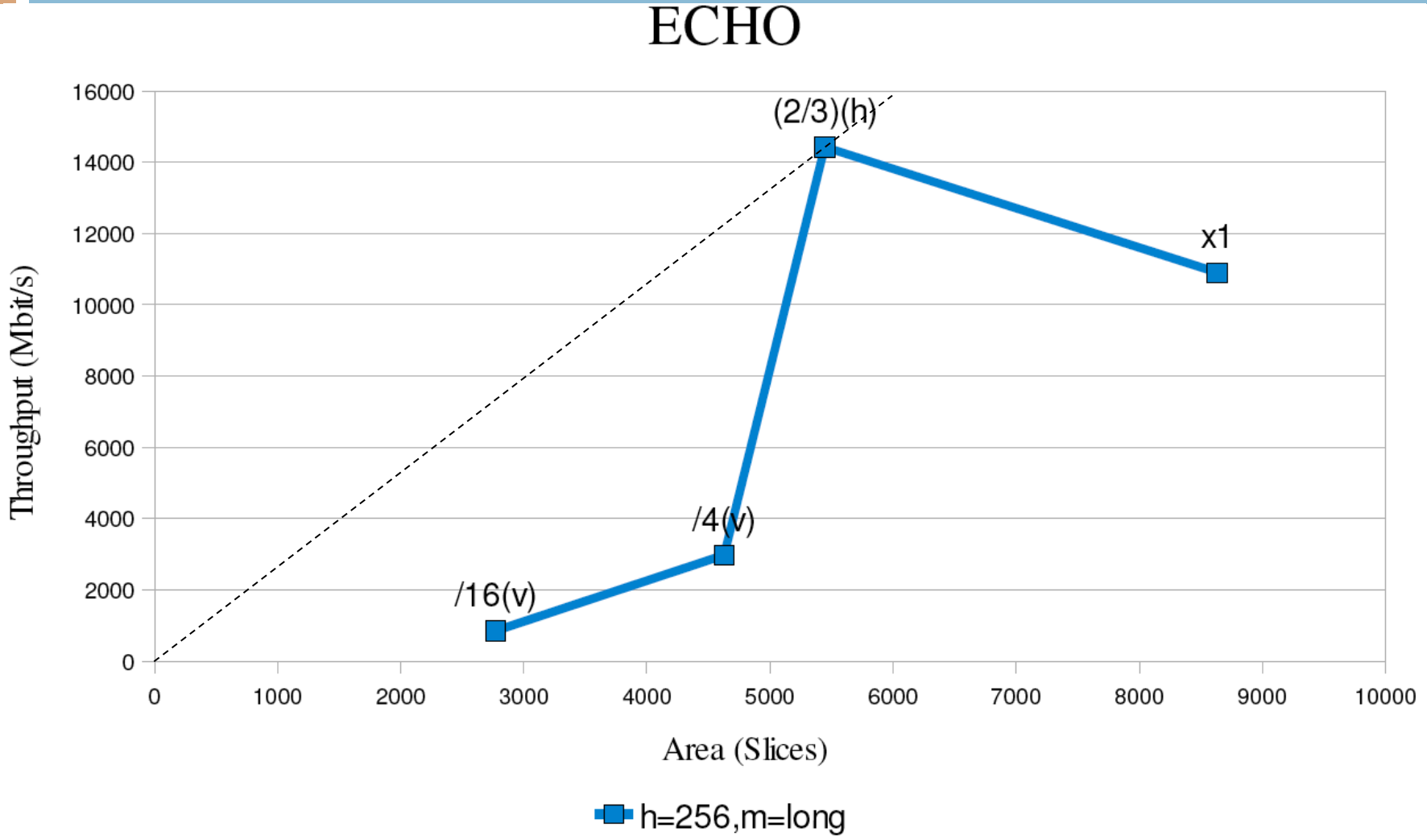
Folding of Luffa



/N(v) – folded vertically by a factor of N xN – unrolled by a factor of N

x1 – basic iterative

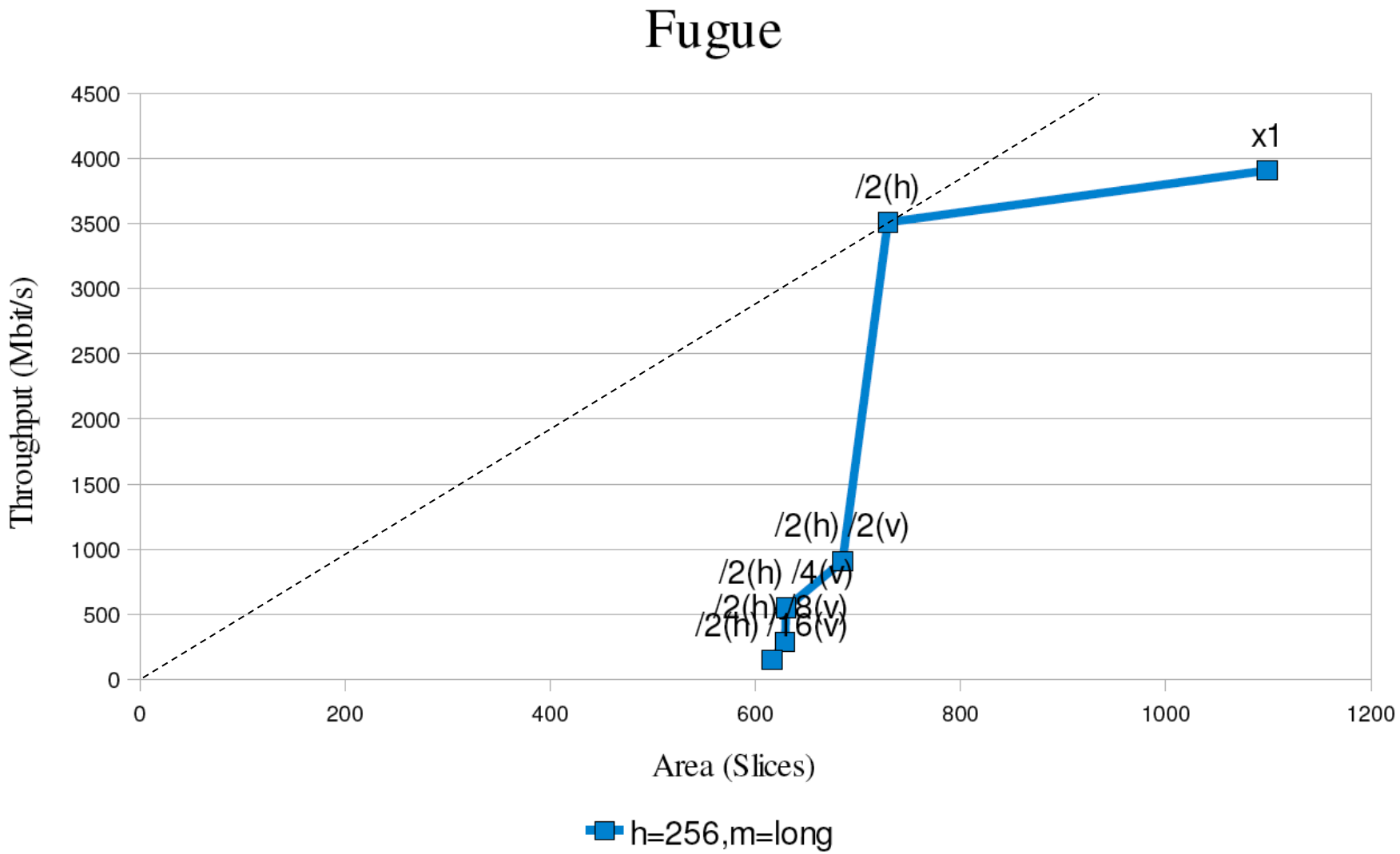
Folding of ECHO



/N(v) – folded vertically by a factor of N x1 – basic iterative

x2/3(h) - folded horizontally by a factor of 3/2 [BIG.SubBytes logic reused]

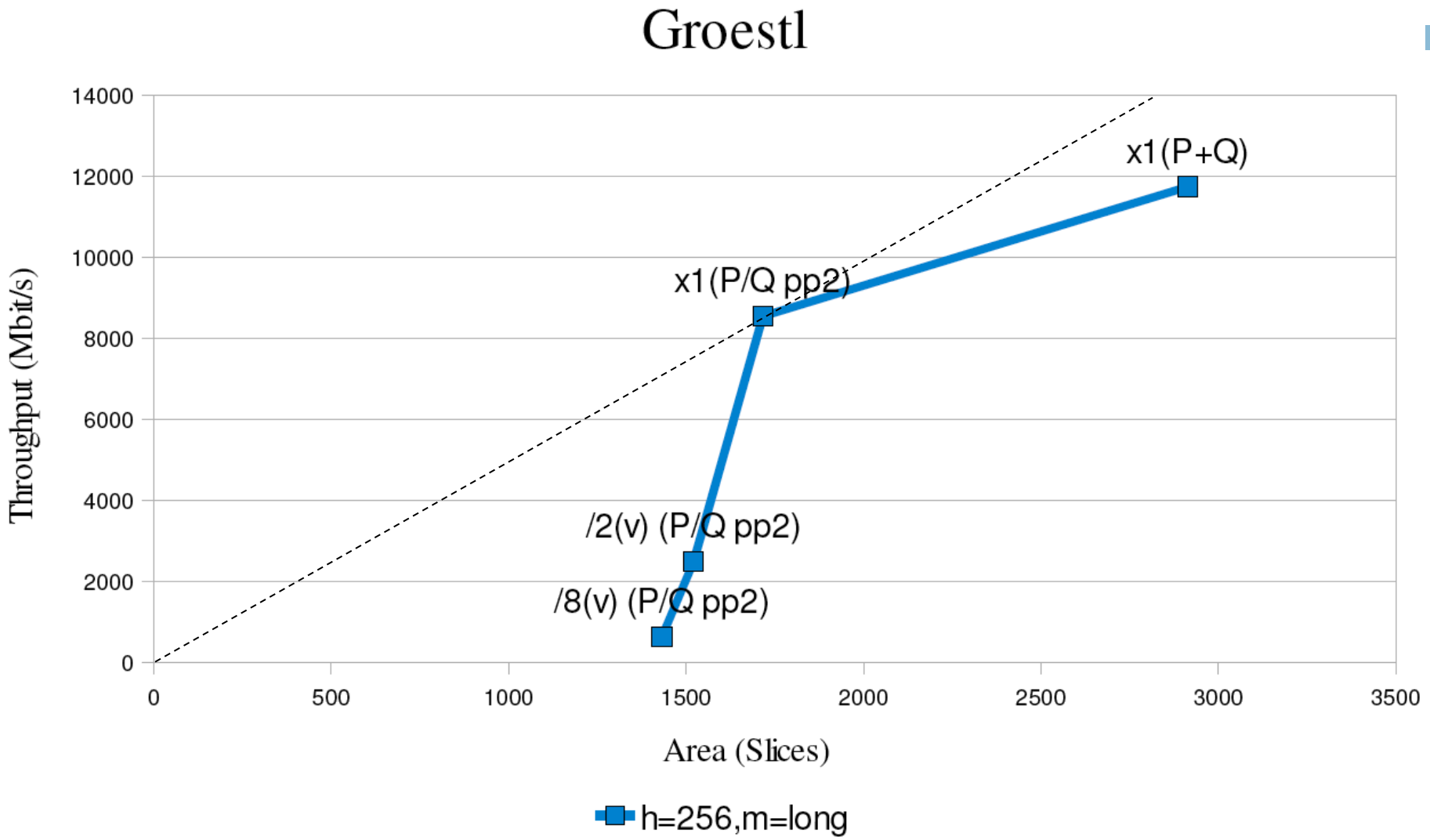
Folding of Fugue



x1 – basic iterative /2(h) – folded horizontally by a factor of 2

/2(h) /N(v) – folded horizontally by a factor of 2, and vertically by a factor of N

Folding of Groestl

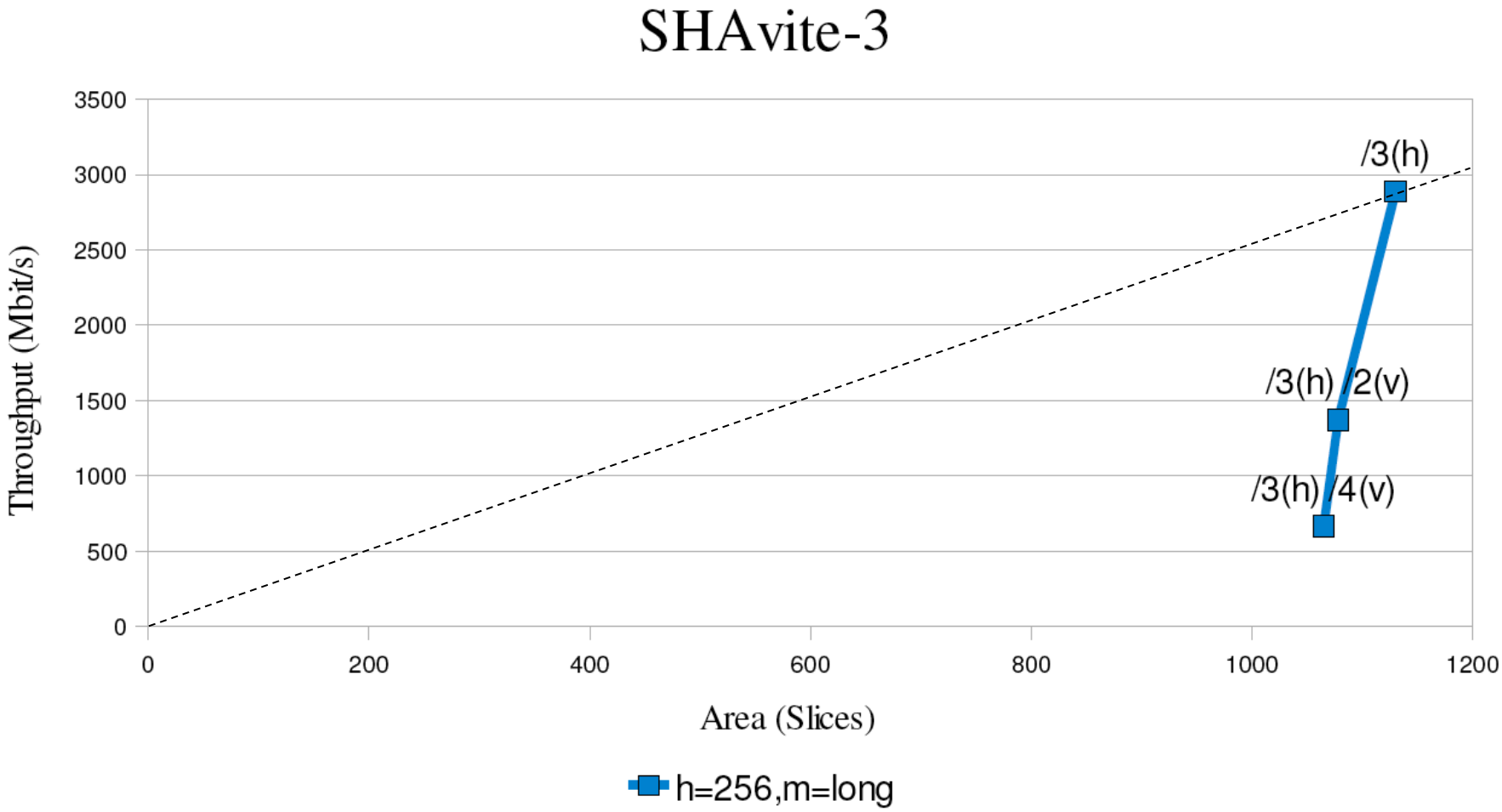


x1(P+Q) – basic iterative with P&Q executing in parallel

x1(P/Q pp2) – basic iterative with P&Q sharing the same logic with two stages of pipelining

/N(v) (P/Q pp2) – folded vertically by a factor of N, shared P/Q with two stages of pipelining

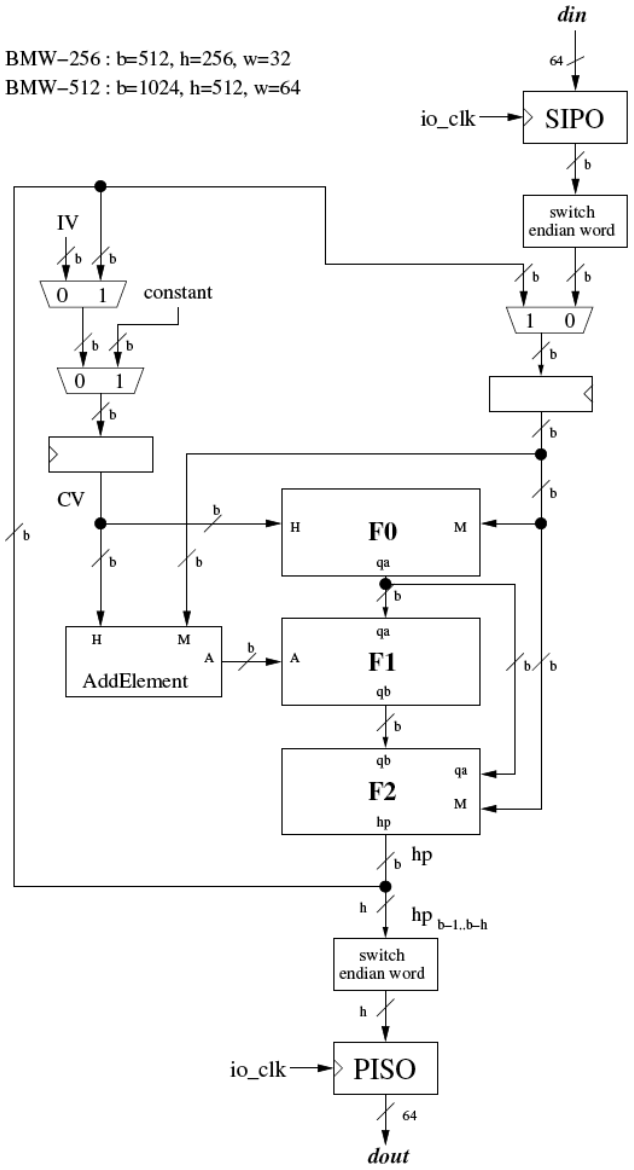
Folding of SHAvite-3



/3(h) – folded horizontally by a factor of 3

/3(h) /N(v) – folded horizontally by a factor of 3, and vertically by a factor of N

Special Case - BMW



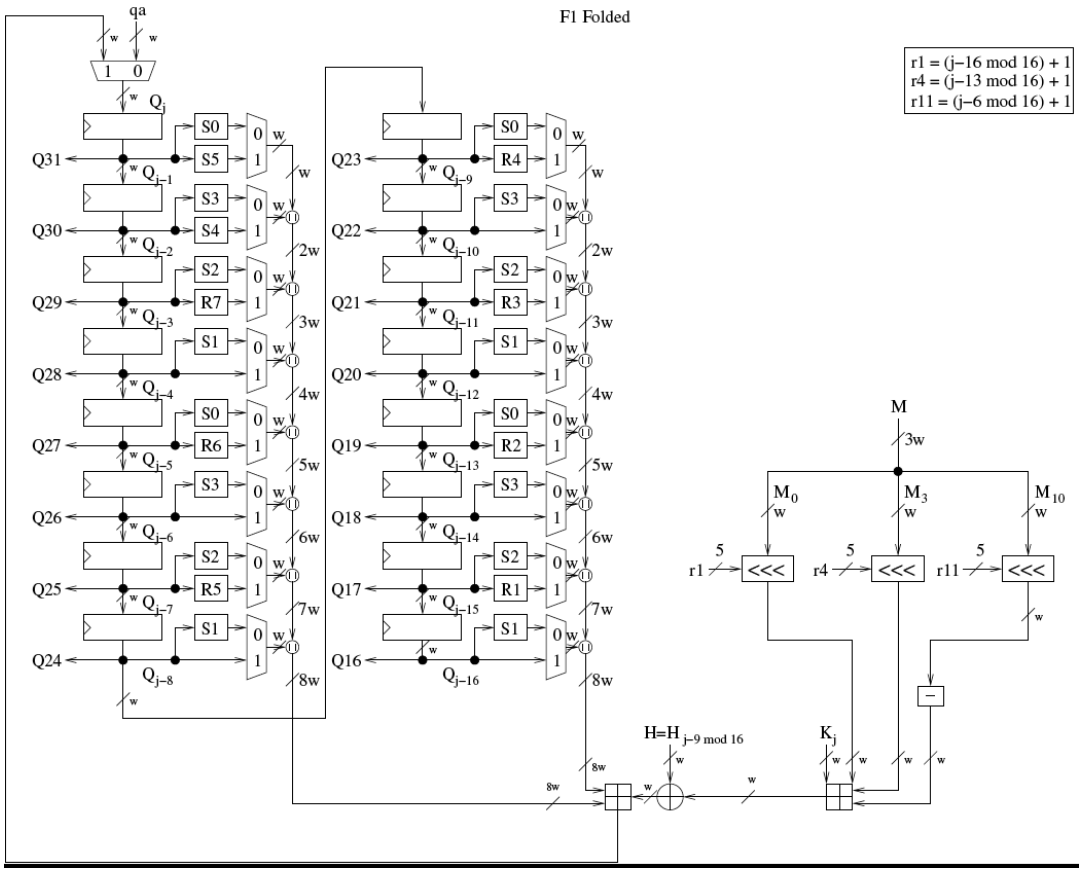
BMW: no clear round structure

Basic architecture combinational

Very large area

One clock cycle per message block

New Folded Architecture of BMW

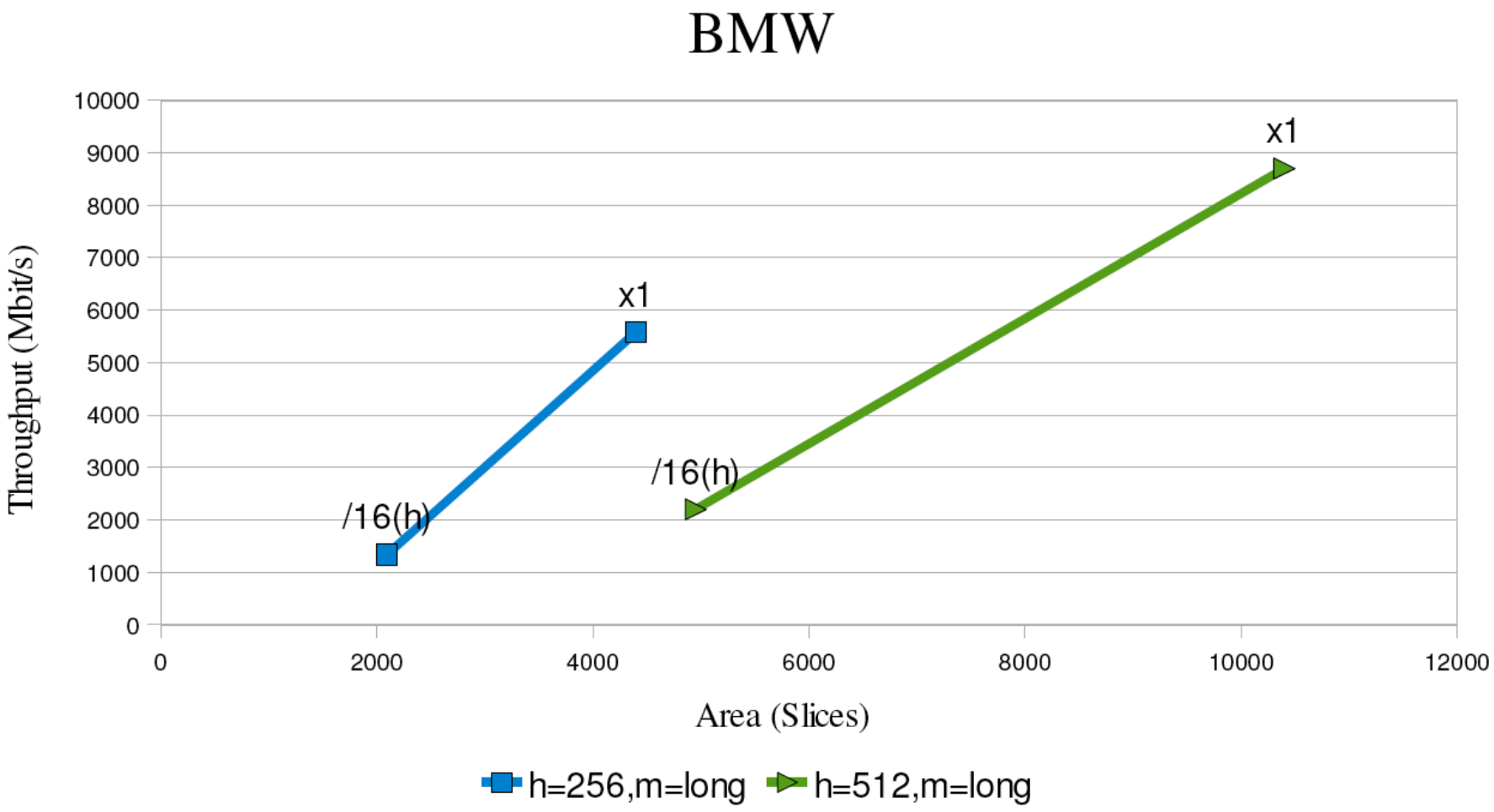


New, previously not reported, folded architecture

33 clock cycles per message block

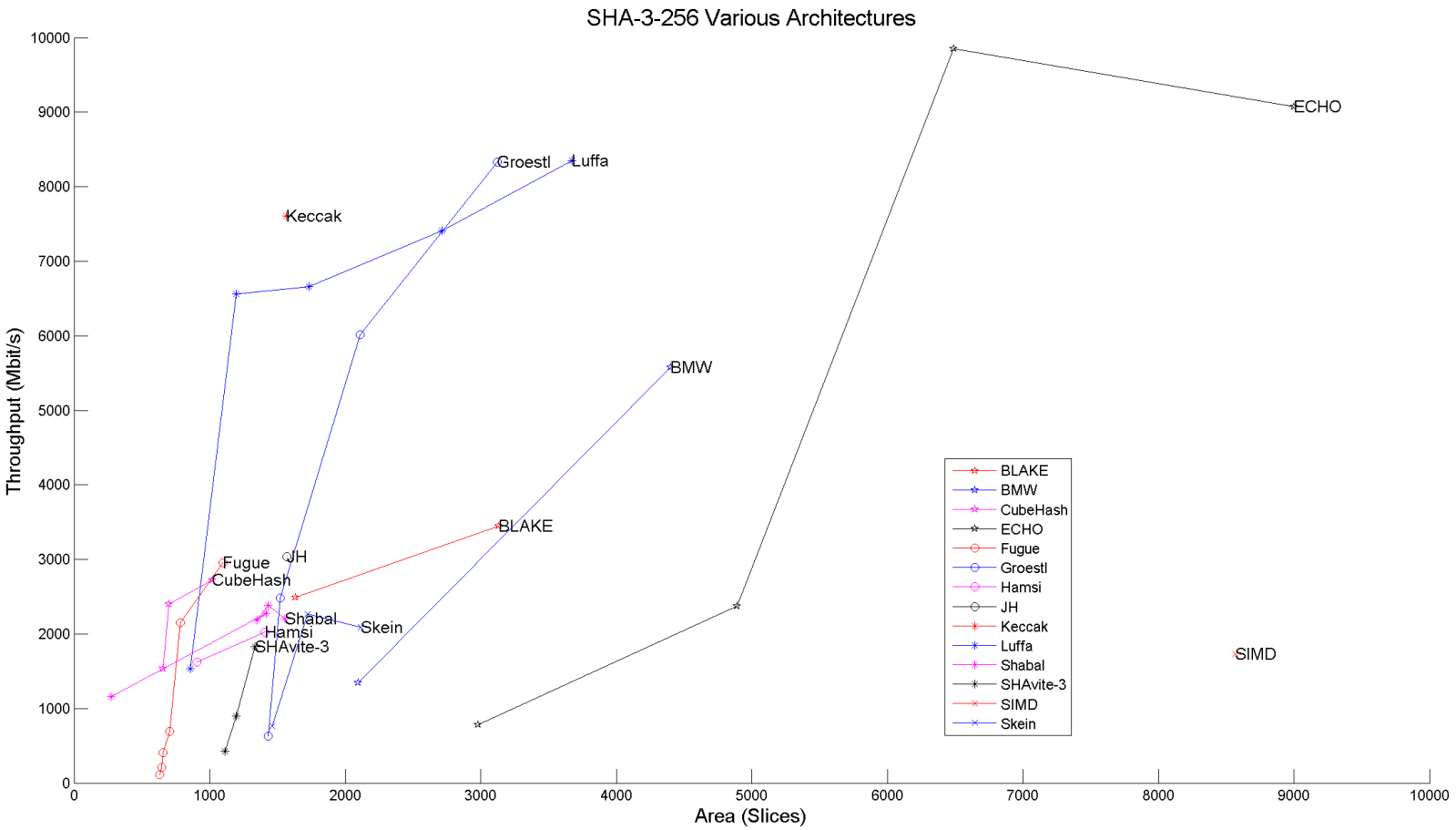
Significant reduction in the circuit area

Special Case - BMW

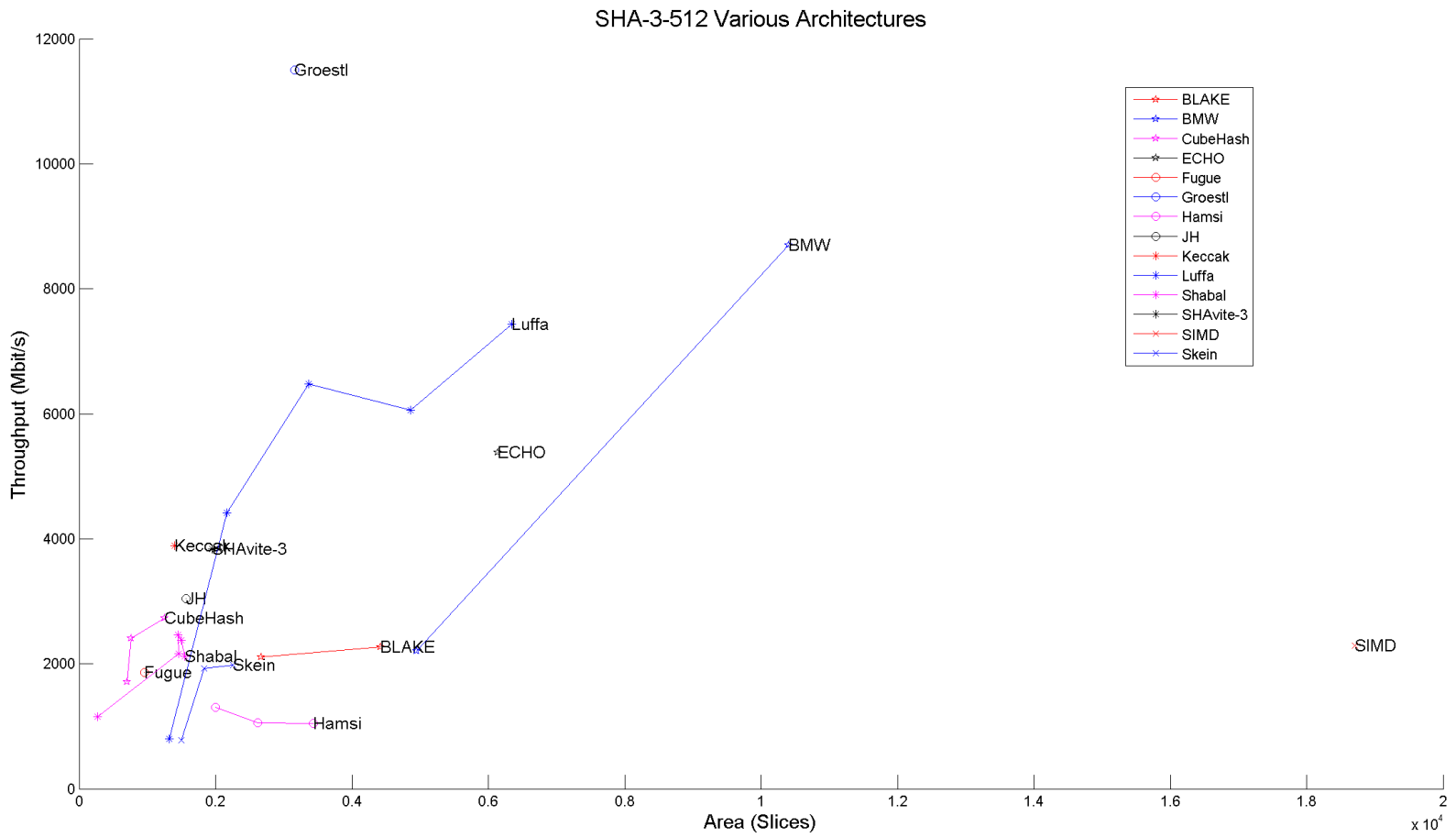


x1 – basic iterative /16(h) – f1 folded horizontally by a factor of 16

Combined Results for 256-bit SHA-3 Variants



Combined Results for 512-bit SHA-3 Variants



Algorithms Ranked According to the Flexibility (1)

Highest Flexibility, Best Area Reduction Factors

BLAKE:	x1, /2(h), /4(h),	/2, /4 (v)
Luffa-256:	x1	/3 (v)
Luffa-512:	x1	/5 (v)

High Flexibility, Medium Area Reduction Factors

ECHO:	x1, x2/3	/2, /4, /8, /16 (v)
Fugue-256:	x1, /2(h),	/2, /4, /8, /16 (v)
Groestl:	x1(P+Q), x1(P/Q pp2)	/2, /4, /8 (P/Q pp2) (v)
JH:	x1	/2, /4, /8, /16, /32, /64 (v)

Algorithms Ranked According to the Flexibility (2)

Moderate Flexibility

BMW:	x1	/16(h)
CubeHash:	x1	/2(h)
SHAvite-3-256:	/3(h)	/2, /4 (v)
SHAvite-3-512:	/4(h)	/2, /4 (v)
Skein:	x1, x4, x8	
Shabal:	x1, x2, x3, x4, x6	

Unknown, Most Likely Low Flexibility

Keccak

Hamsi

How to Increase the Speed? : The case for pipelining and parallel processing

- Protocols: IPSec, SSL, WLAN (802.11)

- Minimum Required Throughput Range: 100 Mbit/s - 40 Gbit/s (based on the specs of Security Processors from Cavium Networks, HiFn, and Broadcom)

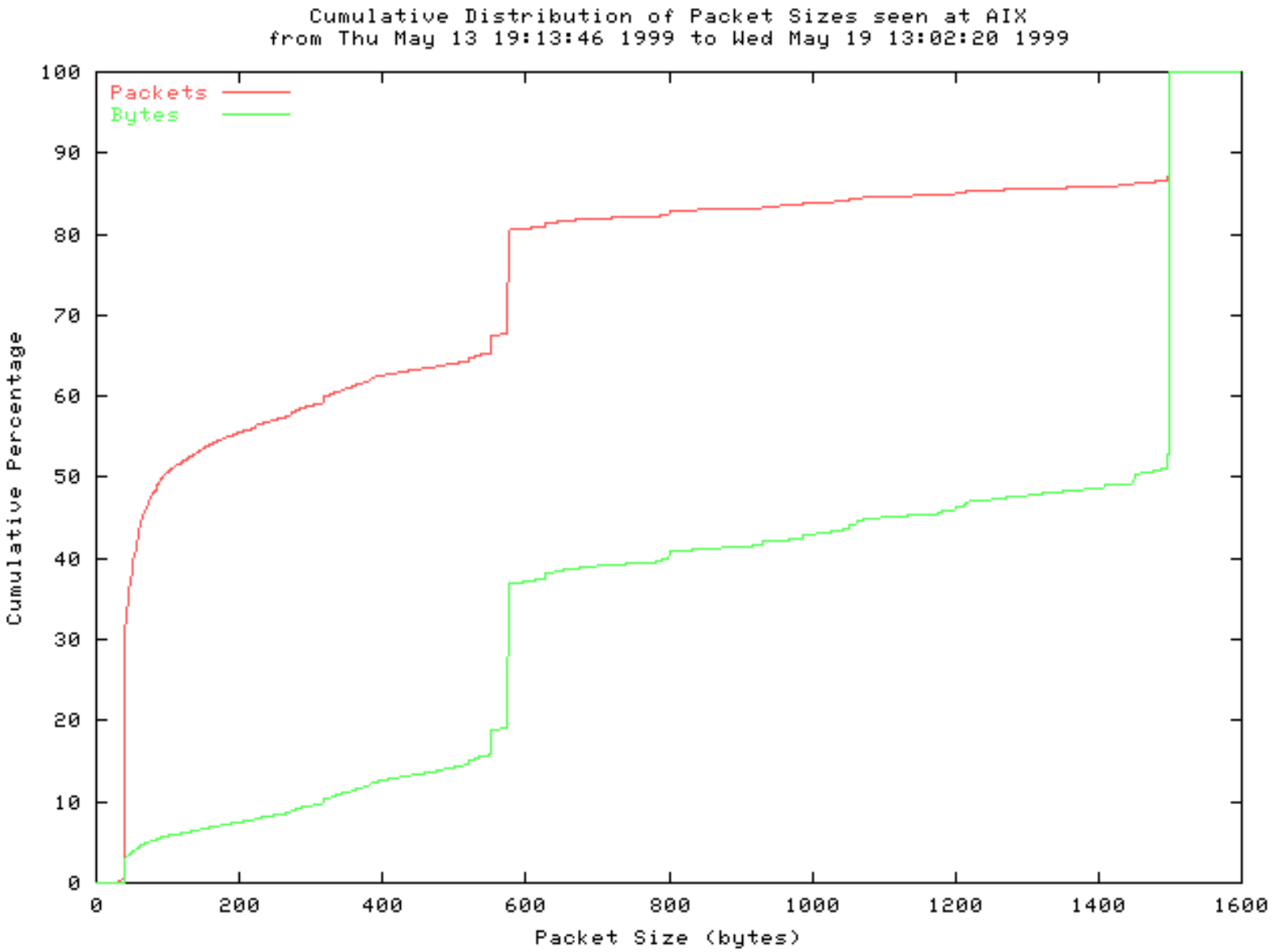
- Supported sizes of packets: 40B - 1500B

  1500 B = Maximum Transmission Unit (MTU) for Ethernet v2

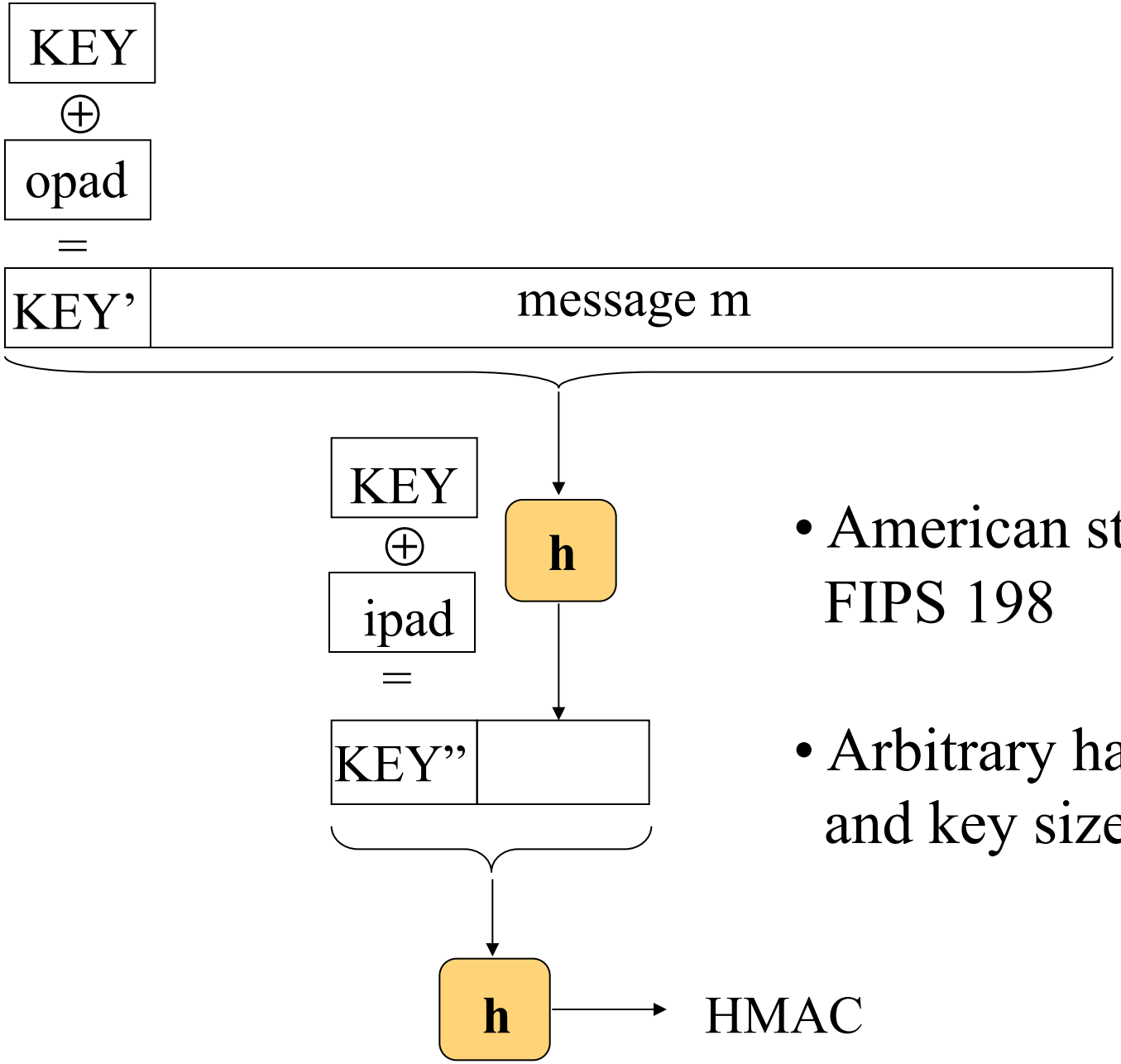
  576 B = Maximum Transmission Unit (MTU) for Internet IPv4 Path

- Most Common Operation Involving Hashing: HMAC

Cumulative Distribution of Packet Sizes

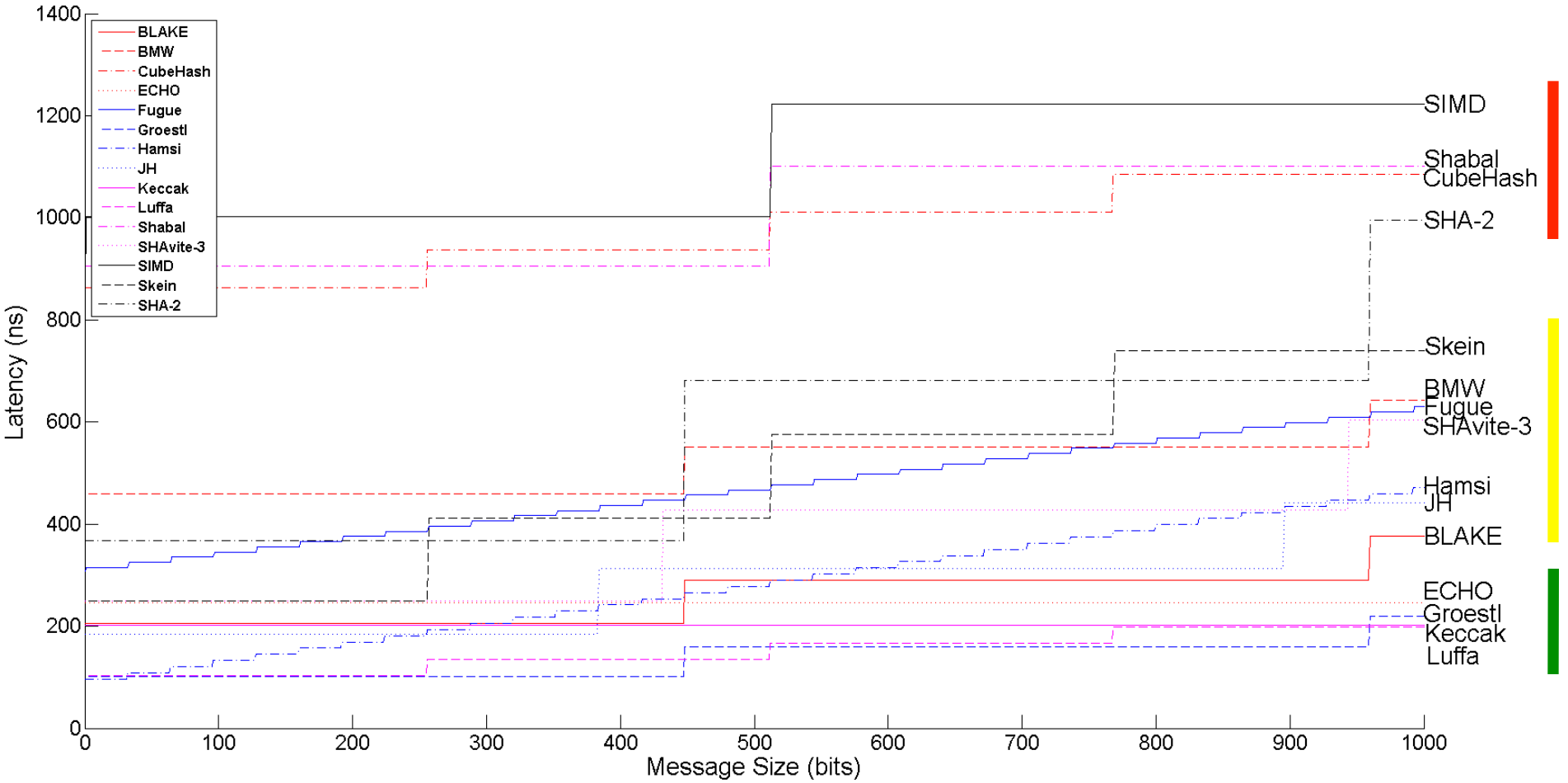


HMAC

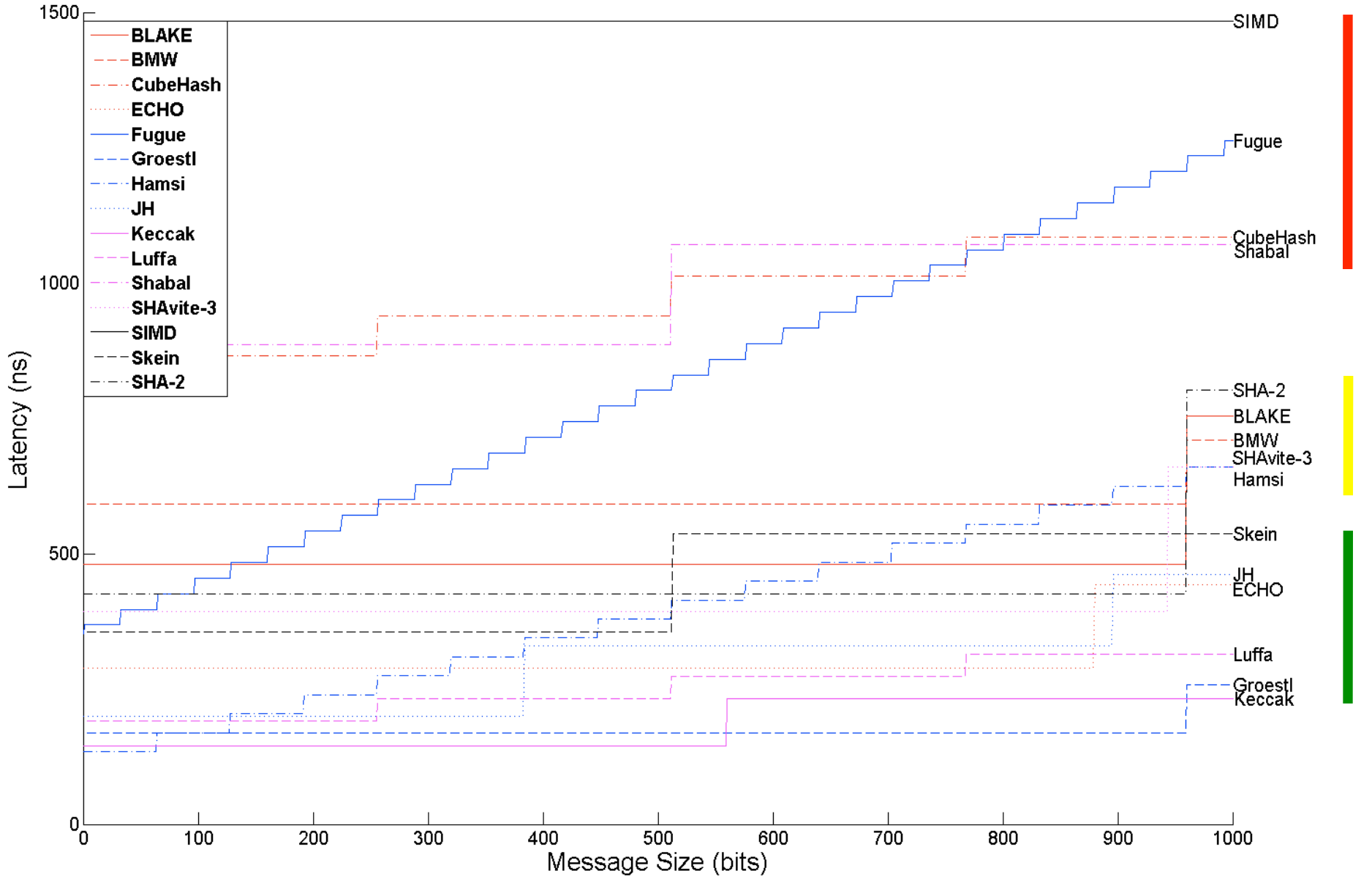


- American standard FIPS 198
- Arbitrary hash function and key size

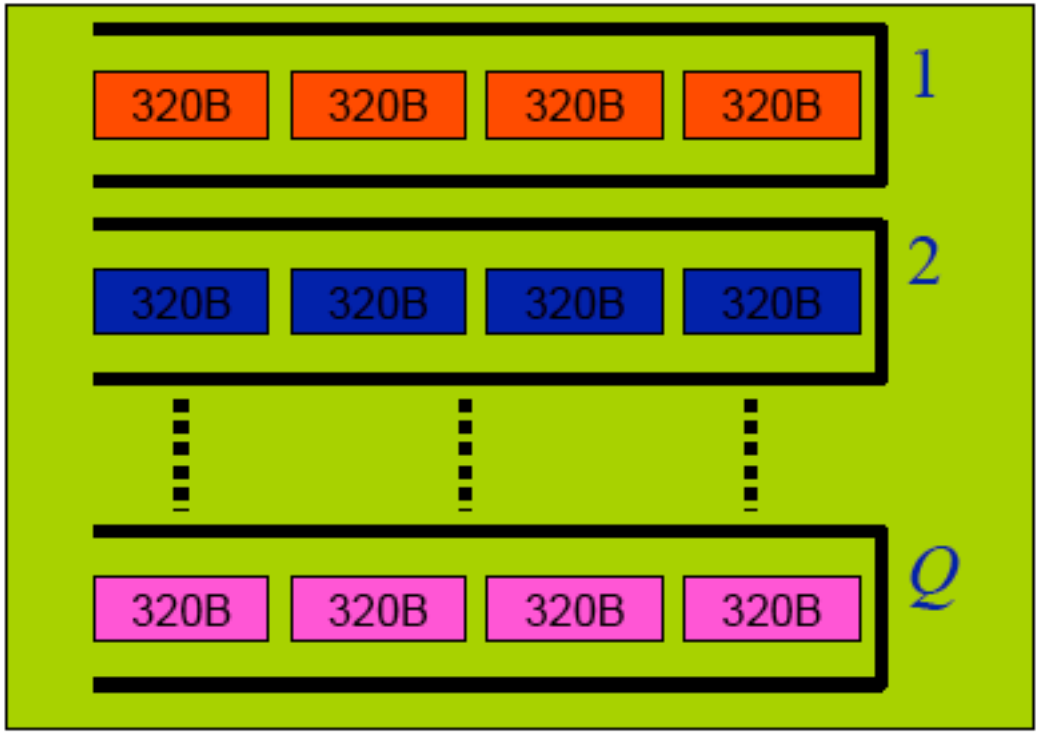
Execution Time for Short Messages up to 1000 bits
Virtex 5, 256-bit variants of algorithms



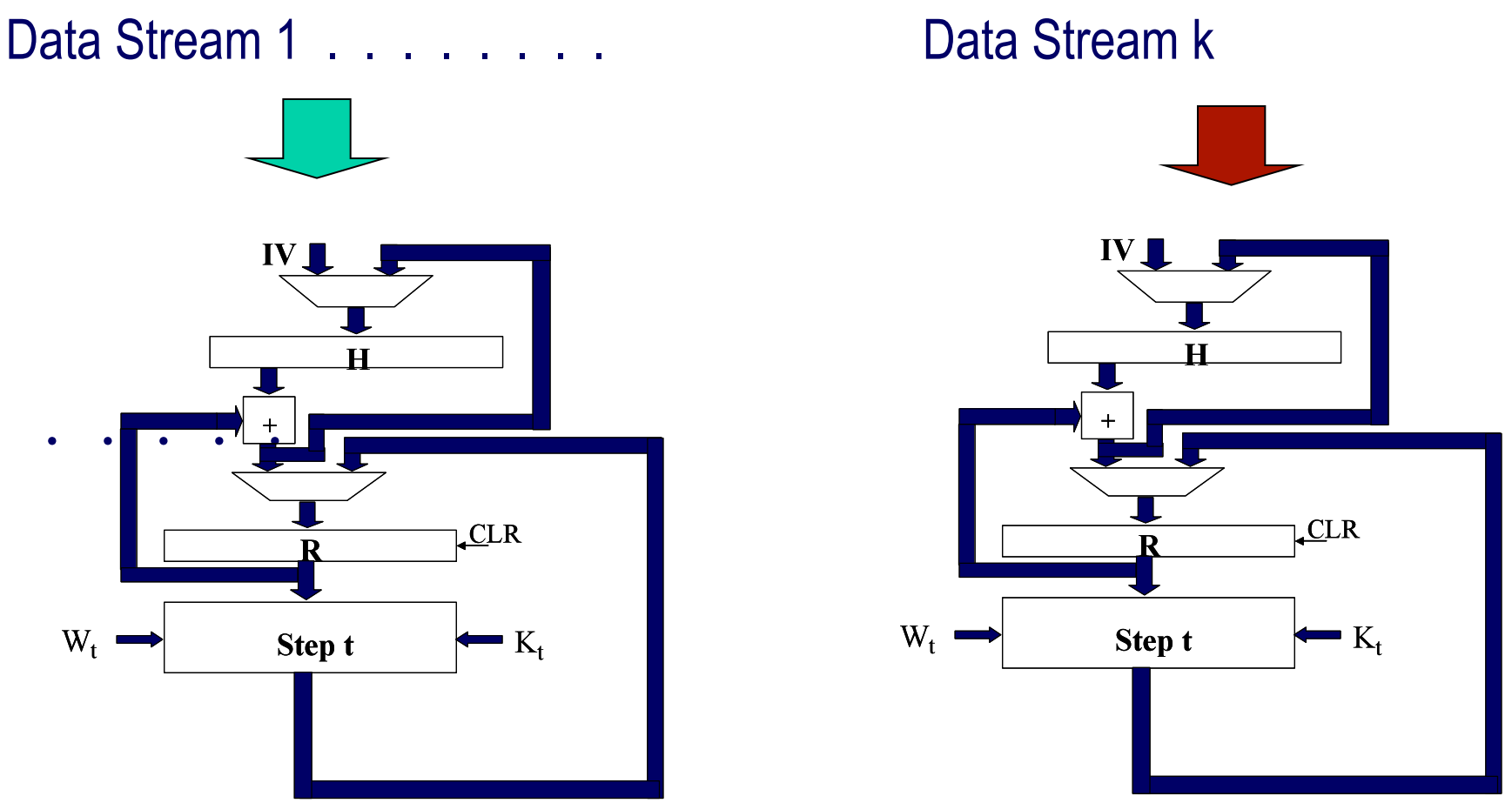
Execution Time for Short Messages up to 1000 bits
Virtex 5, 512-bit variants of algorithms



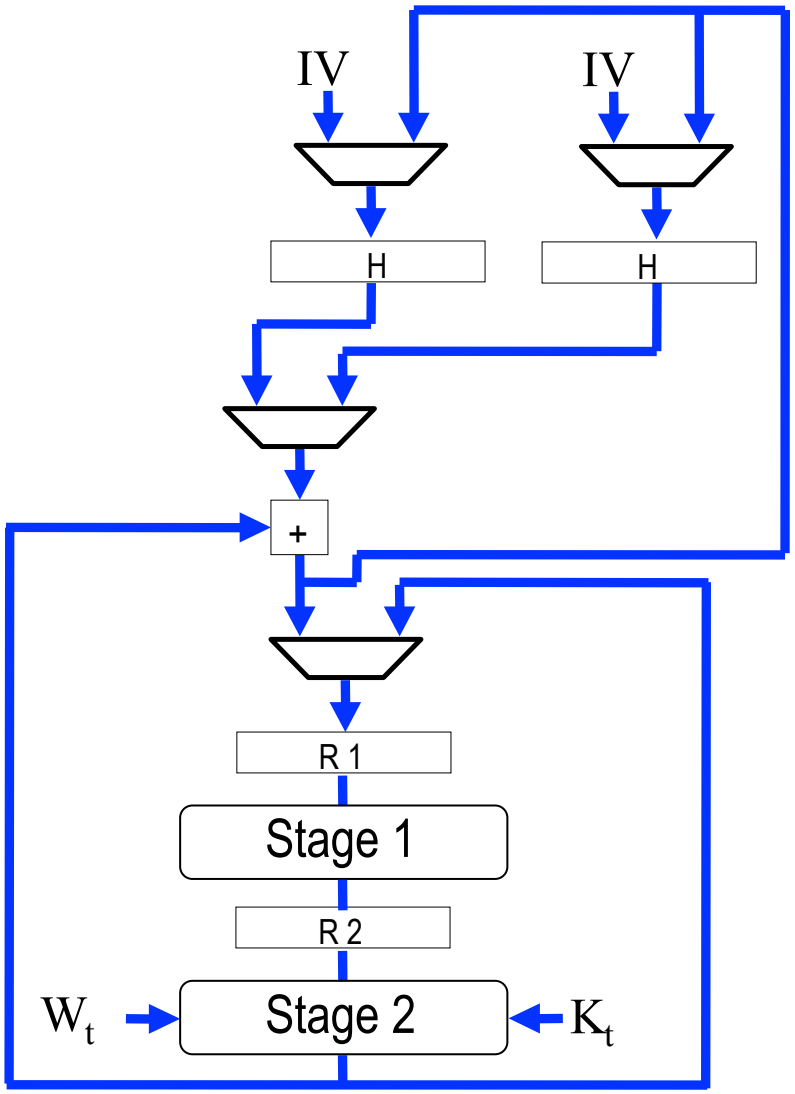
Multiple Packets Available for Parallel Processing



Parallel Processing



Pipelining



BMW vs. CubeHash (1)

In Virtex 5:

Clock period:

CubeHash - 5 ns

BMW - 100 ns

Let us assume that the pipeline can be inserted every 5 ns (clock frequency = 200 MHz).

Number of pipeline stages:

CubeHash - 1

BMW - 20

BMW vs. CubeHash (2)

BMW:

Before pipelining:

Throughput = 512/(100ns) = 5.12 Gbit/s

Area = 4400 CLB slices

After pipelining:

Throughput' = 512/(5ns) = **100 Gbit/s**

Area' = 4400+20*0.1*4400 CLB slices = **13,200 CLB slices**

(assuming 10% increase in area per pipeline stage)

CubeHash:

Throughput = 256/(16*5ns)= 3.2 Gbit/s

Area = 700 CLB slices

In order to reach the speed of **100 Gbit/s**,

the required area of CubeHash = 700 * (100/3.2) = **21,875 CLB slices**

BMW vs. CubeHash (3)

50 Gbit/s

BMW:

N=10

Throughput' = 512/(10ns) = 51.2 Gbit/s

Area' = 4400+10*0.1*4400 CLB slices = **8,800 CLB slices**

CubeHash:

In order to reach the speed of 50 Gbit/s:

The required area of CubeHash = 700 * (50/3.2) = **10,938 CLB slices**

Reported Pipelined Implementations by Savas et al.

Pipelined Implementations:

Multi-Message Hashing by Savas et al.

Number of pipeline stages

Keccak - 5

Luffa - 2

BMW - 18

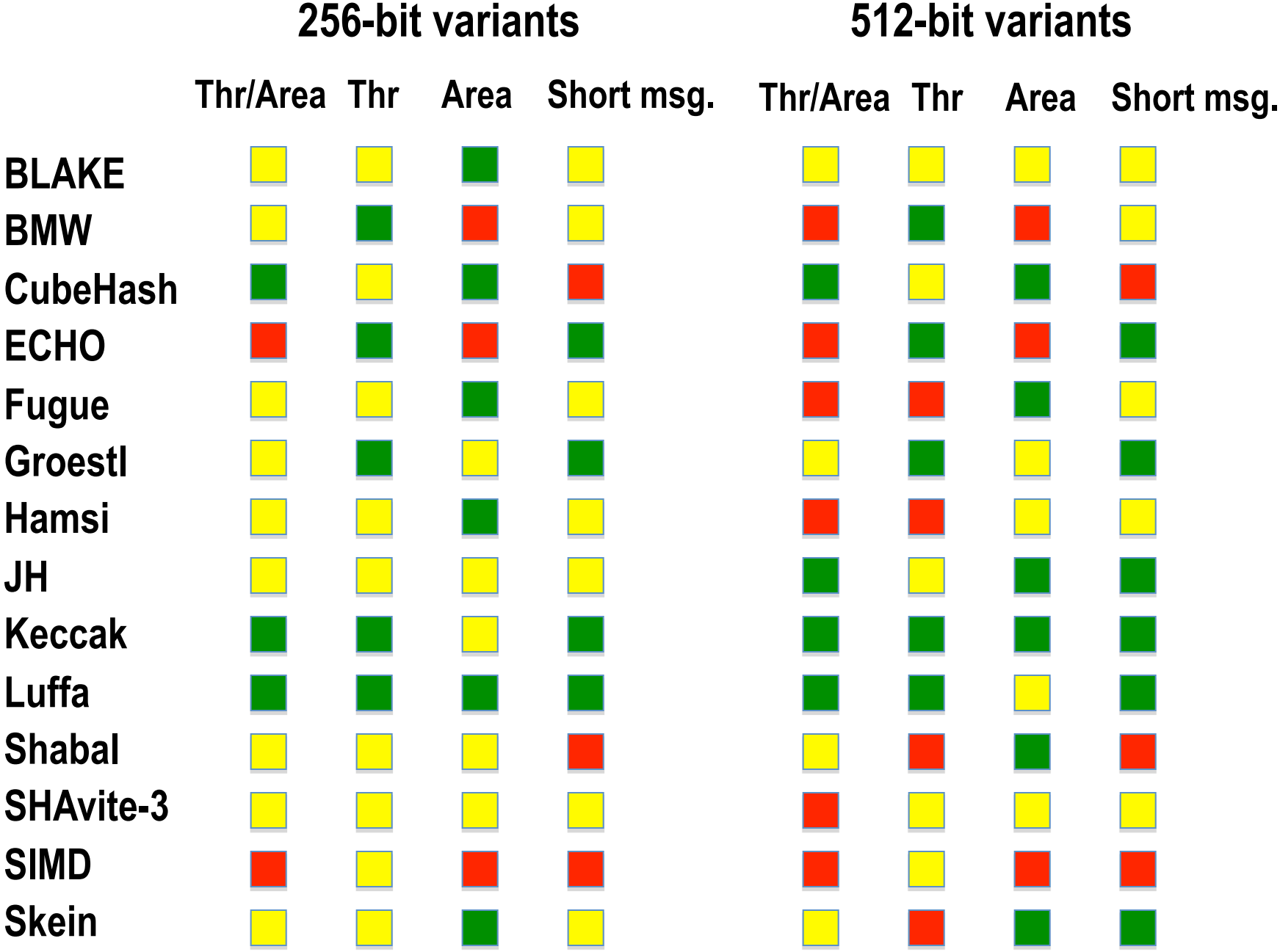
Results for Spartan3, Virtex 2, Virtex 4, 90nm ASIC.

Improvement of the Throughput/Area ratio

	Spartan 3	Virtex 2	Virtex 4	ASIC 90nm
Keccak	2.0	1.2	1.7	1.7
Luffa	1.1	1.2	1.3	1.5
BMW	11.5	11.0	10.8	1.8

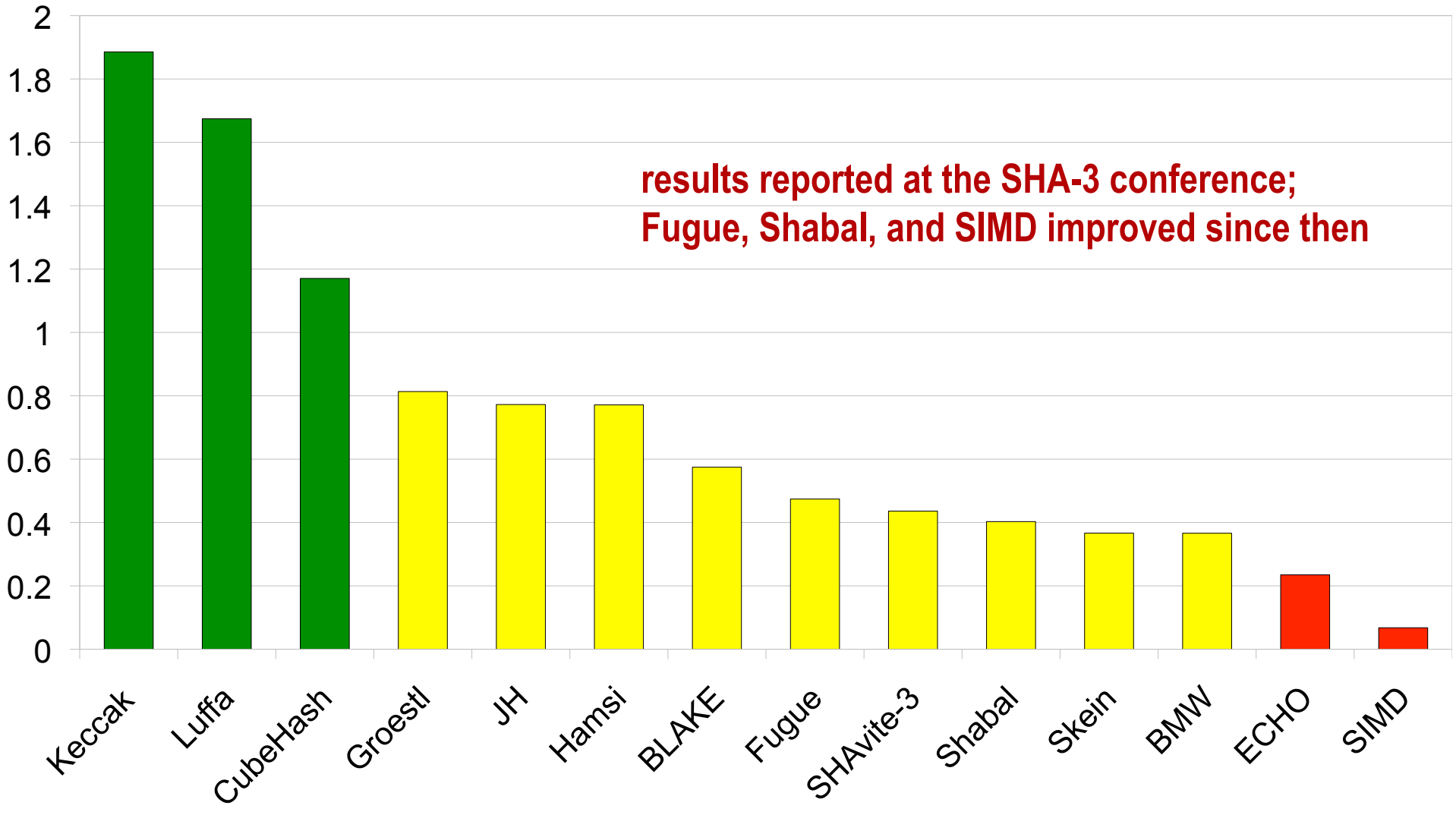
Maximum Throughputs [Gbit/s] Reached

	Spartan 3	Virtex 2	Virtex 4	ASIC 90nm
Keccak	14.9	15.0	22.3	74.4
Luffa	6.0	12.5	13.9	35.4
BMW	28.7	43.2	58.0	133.0


256-bit variants
512-bit variants
Thr/Area
Thr
Area
Short msg.
Thr/Area
Thr
Area
Short msg.
BLAKE
BMW
CubeHash
ECHO
Fugue
Groestl
Hamsi
JH
Keccak
Luffa
Shabal
SHAvite-3
SIMD
Skein

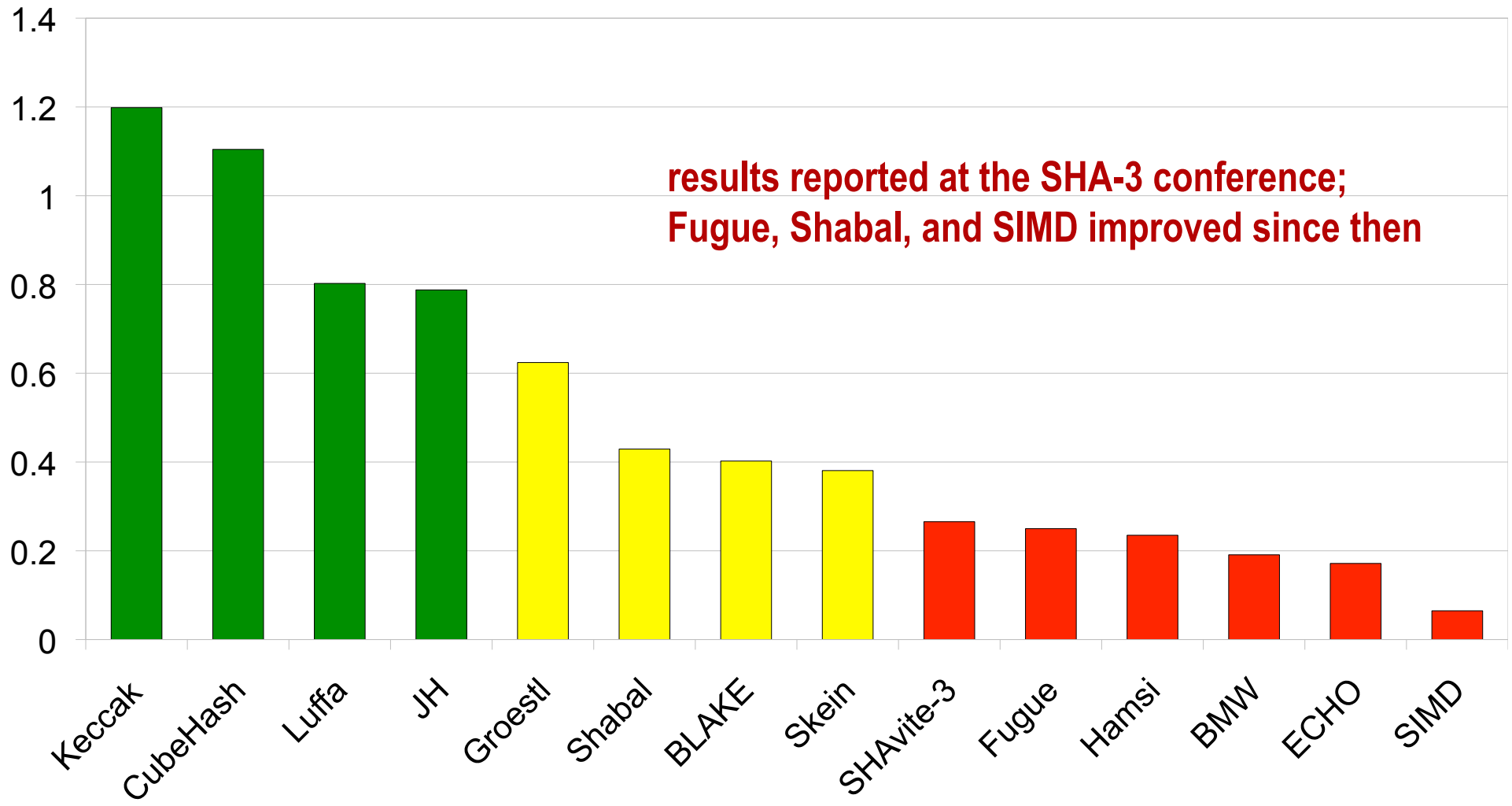
Overall Normalized Throughput/Area: 256-bit variants

Normalized to SHA-256, Averaged over 7 FPGA families

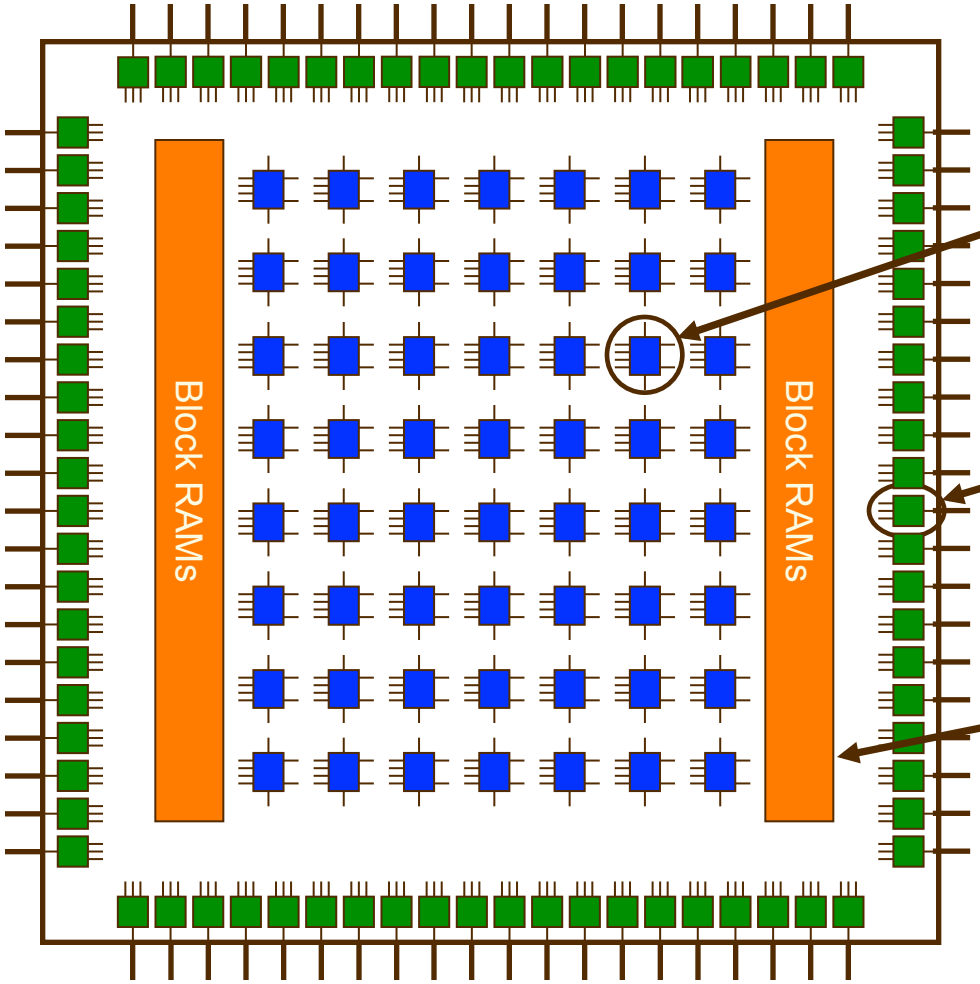


Overall Normalized Throughput/Area: 512-bit variants

Normalized to SHA-512, Averaged over 7 FPGA families



What is an FPGA?

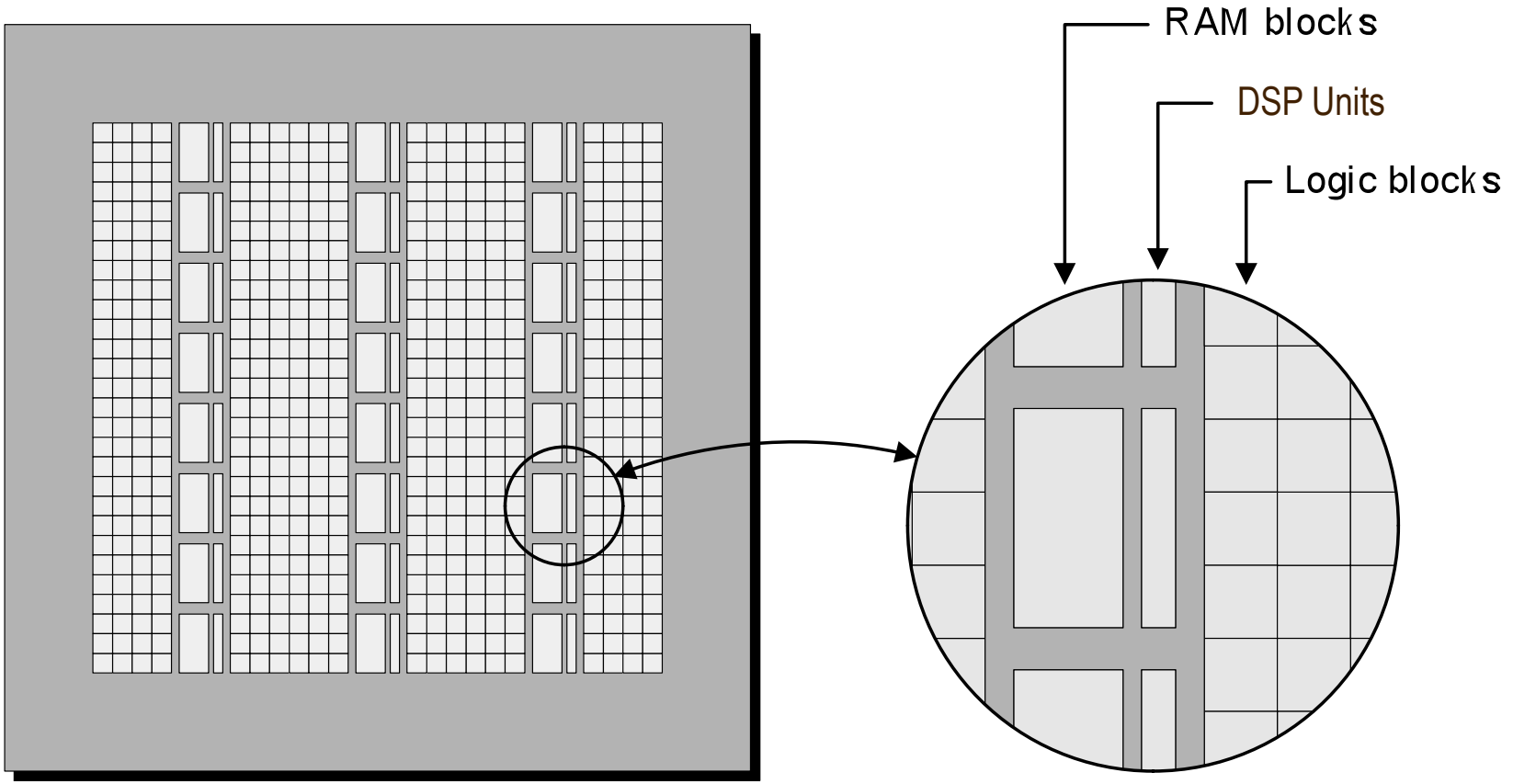


Configurable Logic Blocks

I/O Blocks

Block RAMs

RAM Blocks and DSP Units In Xilinx and Altera FPGAs



The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043

Hash Algorithm	DSP Adders	DSP Multipliers	Block Memories
BLAKE	Yes	-	Yes
BMW	Yes	-	-
CubeHash	Yes	-	-
ECHO	-	-	Yes
Fugue	-	-	Yes
Groestl	-	-	Yes
Hamsi	-	-	Yes
JH	-	-	Yes
Keccak	-	-	Yes
Luffa	-	-	-
SHA-2	Yes	-	Yes
Shabal	Yes	Yes	-
SHAvite-3	-	-	Yes
SIMD	Yes	Yes	Yes
Skein	Yes	-	-

BLOCK MEMORIES

Block Memories used to implement T-boxes/S-boxes

- ECHO, SHAvite-3
  - AES-Sboxes (8x8)
  - AES-Tboxes (8x32)
- Fugue
  - AES-Sboxes (8x8)
  - Fugue-Tboxes (8x24) and (8x32)
- Groestl
  - AES-Sboxes (8x8)
  - Groestl-Tboxes (8x40)

Block Memories used to implement ROM and Round Constants

- Hamsi
  - ROM in message expansion
    8x4x256x32 = 256 kbit in Hamsi-256
- Keccak, JH, SHA-2
  - Round constants only
- BLAKE
  - Permutation

DSP ADDERS
&
MULTIPLIERS

DSP Adders

- CubeHash
  - 32-bit addition
- Skein
  - 64-bit addition
- BMW
  - 32-bit or 64-bit Multioperand Addition
- BLAKE
  - 32-bit Addition
- SHA-2
  - 32-bit Multioperand Addition

PRELIMINARY RESULTS

DSP Adders & Multipliers

Algorithm	Architecture	Max Clk Freq [Mhz]	Throughput [Mbits/s]		Area [CLB Slices, BRAM, DSP]
CubeHash	Basic	215.33	3445.00	✔	707, 0, 0
	Embedded	234.41	3751.00		706, 0, 32
Skein	Basic	104.34	2812.00	✗	1463, 0, 0
	Embedded	50.55	1362.15		1111, 0,64
BMW	Basic	10.89	5576.70	✗	4400, 0, 0
	Embedded	5.29	2707.50		3267, 0, 144
Shabal	Basic	144.47	2958.68	✔	275, 0, 0
	Embedded	208.55	4271.00		298, 0, 70
SIMD	Basic	40.89	2325.90	✗	9288, 0, 0
	Embedded	33.73	1919		7880, 0, 96

✔ - Throughput increases

✗ - Throughput decreases
(most likely as a result of design error)

Block Memory & Adders

Algorithm	Architecture	Max Clk Freq [Mhz]	Throughput [Mbits/s]		Area [CLB Slices, BRAM, DSP]
Hamsi	Basic	248.08	2646.00	✗	946, 0, 0
	Embedded	224.07	2390.02		549, 32,0
Keccak	Basic	238.38	10807.00	✔	1229, 0, 0
	Embedded	242.94	10952.70		1320, 2,0
SHA-2	Basic	207.00	1630.00	✗	433, 0, 0
	Embedded	169.12	1332.00		291, 1, 11
BLAKE	Basic	117.06	2853.91	✔	1871, 0, 0
	Embedded	164.42	4008.7		928, 10, 24
JH	Basic	278.09	3955.02		1108, 0, 0
	Embedded				

✔ - Throughput increases

✗ - Throughput decreases
(most likely as a result of design error)

Results by Other Groups: Comprehensive Comparisons

Baldwin et al.

Institutions: University College Cork, Ireland
RMIT University, Melbourne, Australia
Queen's University Belfast, Belfast, UK

Presented at: SHA-3 Candidate Conference 2010, FPL 2010

Matsuo et al.

Institutions: National Institute of Information and Communications Technology, Japan
Katholieke Universiteit Leuven, Belgium
Virginia Tech, USA
National Institute of Advanced Industrial Science and Technology, Japan
The University of Electro-Communications, Japan

Presented at: HOST 2010 and SHA-3 Candidate Conference 2010

Results by Other Groups: Comprehensive Comparisons

Guo et al.

Institutions: Virginia Tech, USA

Presented at: ePrint 2010/536

Results by Other Groups: Interesting Studies of Selected Algorithms

Savas et al.

Institutions: Sabanci University, Istanbul, Turkey

Presented at: SHA-3 Candidate Conference 2010

Pipelined Architectures of BMW, Keccak, and Luffa.

Detrey et al.

Institutions: LORIA, INRIA / CNRS / Nancy Universit e / SGDSN / ANSSI, France

Presented at: Selected Areas in Cryptography, SAC 2010

Excellent Implementation of Shabal

Differences: Padding

Padding in hardware:

Baldwin et al.: yes

- under assumption that the message ends on a boundary of a 32-bit word
- counters in the padding unit of the following functions form the critical path and thus affect the maximum clock frequency

Echo-256, Fugue-256/512, JH

Other groups: no

GMU: no results with padding yet

universal padding circuit under development

- arbitrary SHA-3 candidate, SHA-2, SHA-1
- message allowed to end on a boundary of a word, byte, or bit

Differences: Interface (1)

Baldwin et al.:

- 32-bit input bus
- one clock (the same clock for processing and i/o)

As a result, the interface substantially limits the throughput of the following algorithms:

BMW, Echo, Grøstl, Keccak.

Matsuo et al., Guo et al.:

- compatible with SASEBO boards
- 16-bit input bus
- one clock (the same clock for processing and i/o)

As a result, the interface substantially limits the throughput for majority of algorithms.

Differences: Interface (2)

GMU:

- 64-bit input bus (for all algorithms except those with the block size = 32 bits)
- two clocks, if needed to assure that

  Load Time <= Processing Time

  (only BMW for basic architectures)
- the interface does not restrict the speed of processing for any SHA-3 candidate

Differences: Interface (3)

In order to make the comparison fair, we make the following assumptions:

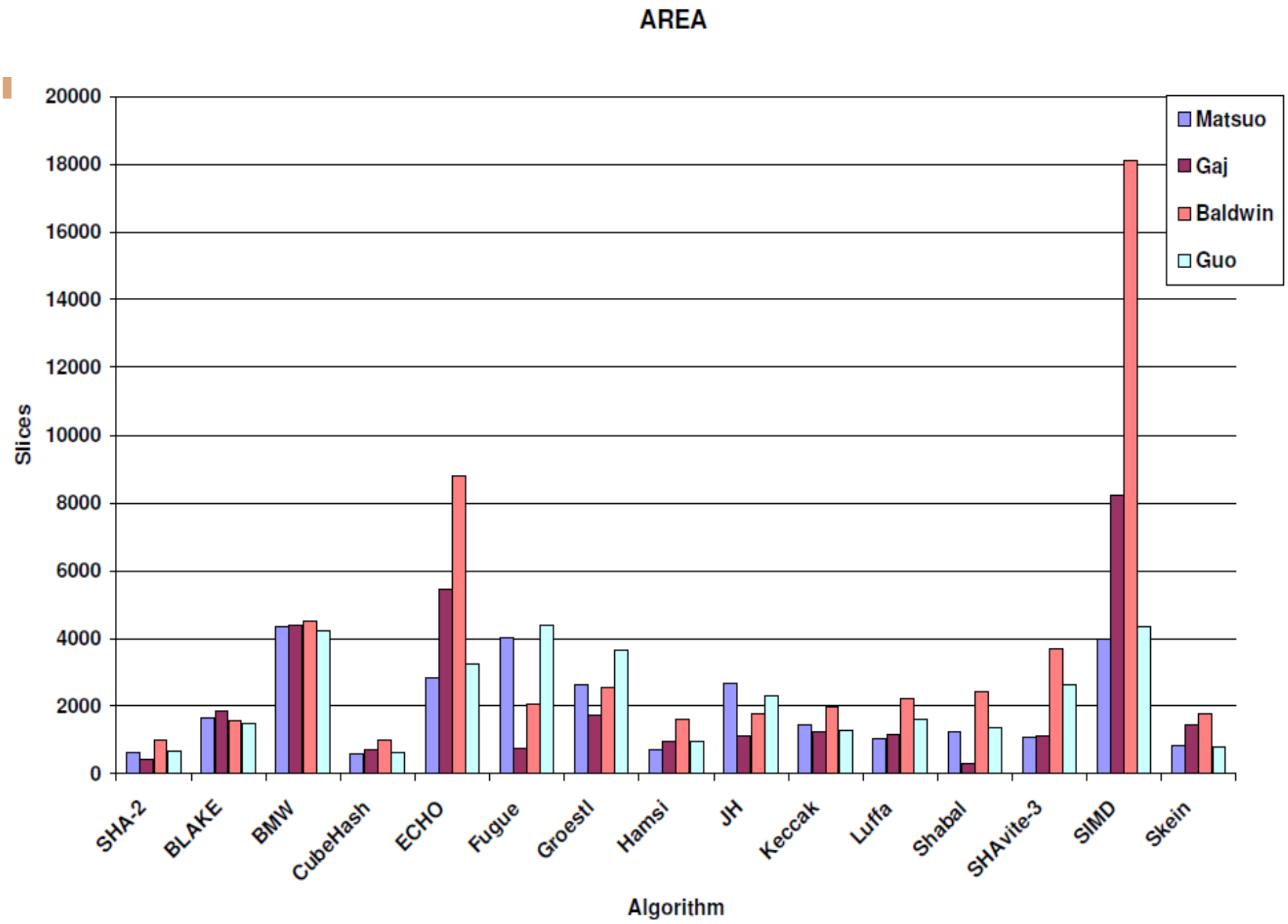
Baldwin et al.

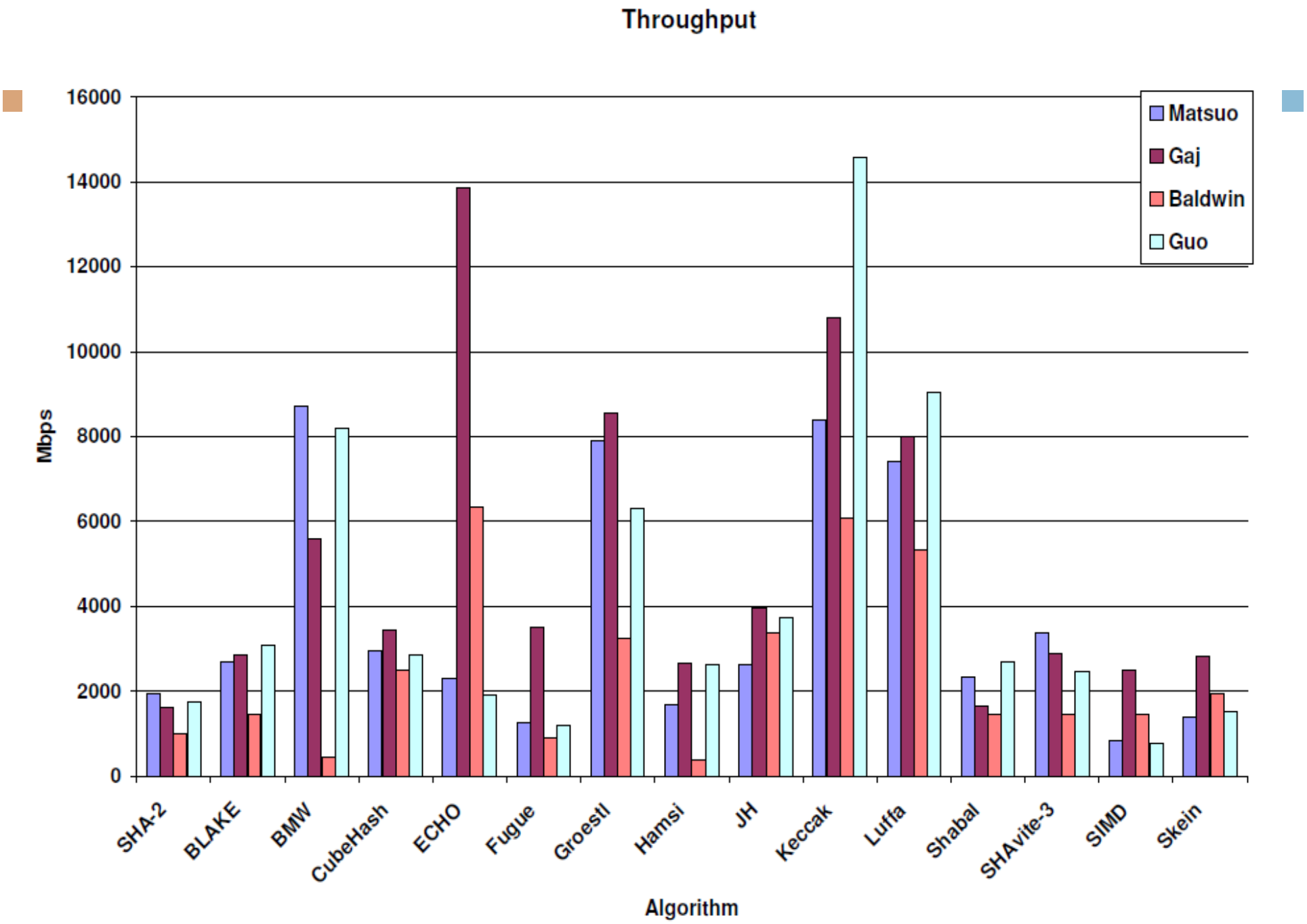
- Padding in Software
- Ideal Input-Output Bus

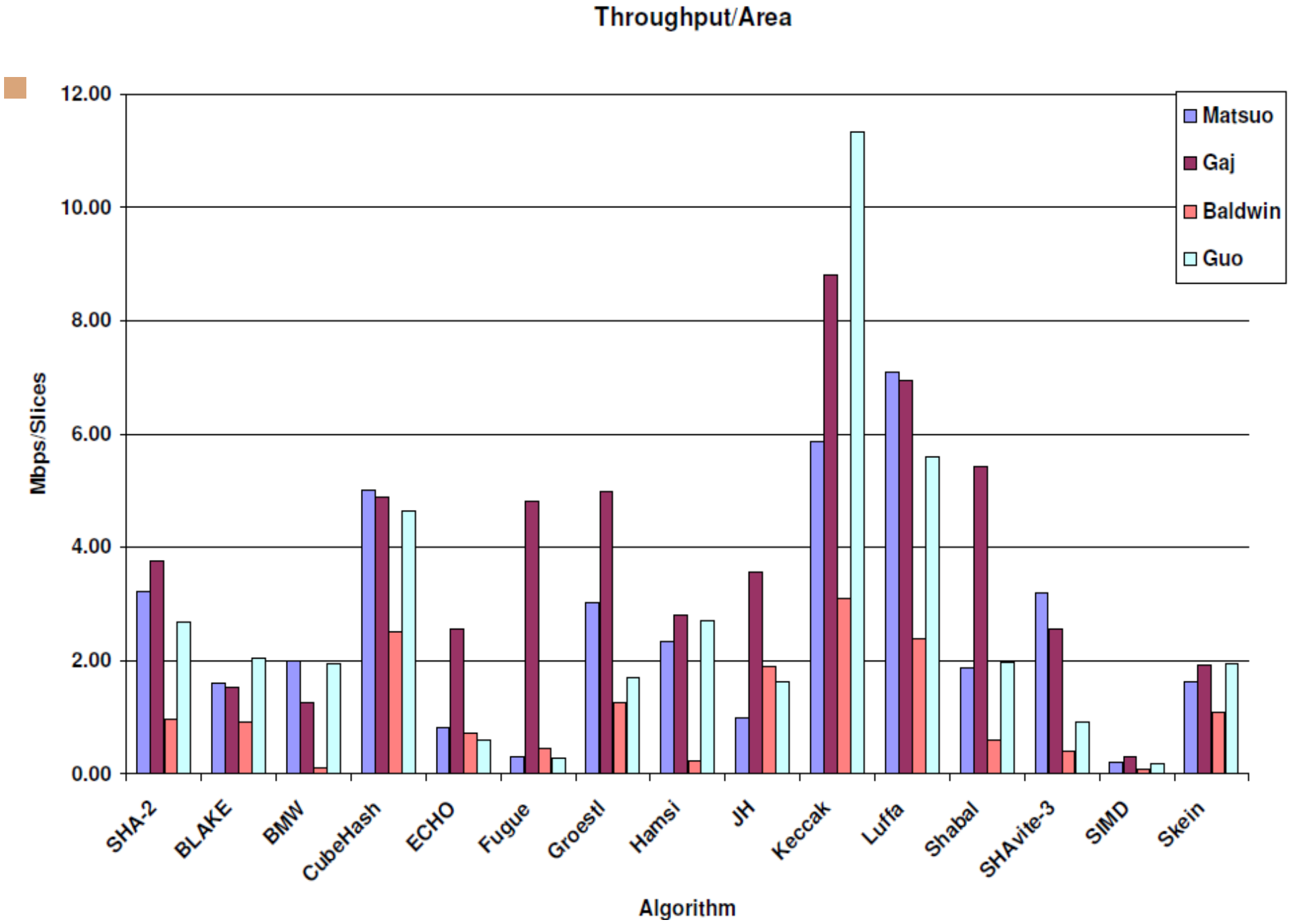
Matsuo et al., Guo et al.

- Ideal Input-Output Bus

Slightly favors other groups in terms of area,
because our interface includes serial-to-parallel and parallel-to-serial converters, as well as message length counters.


AREA
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Slices
Matsuo
Gaj
Baldwin
Guo
SHA-2
BLAKE
BMW
CubeHash
ECHO
Fugue
Groestl
Hamsi
JH
Keccak
Luffa
Shabal
SHAvite-3
SIMD
Skein
Algorithm


Throughput
16000
14000
12000
10000
8000
6000
4000
2000
0
Mbps
Matsuo
Gaj
Baldwin
Guo
SHA-2
BLAKE
BMW
CubeHash
ECHO
Fugue
Groestl
Hamsi
JH
Keccak
Luffa
Shabal
SHAvite-3
SIMD
Skein
Algorithm


Throughput/Area
Mbps/Slices
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Matsuo
Gaj
Baldwin
Guo
SHA-2
BLAKE
BMW
CubeHash
ECHO
Fugue
Groestl
Hamsi
JH
Keccak
Luffa
Shabal
SHAvite-3
SIMD
Skein
Algorithm

Explanation of Remaining Differences
Baldwin et al. vs. GMU

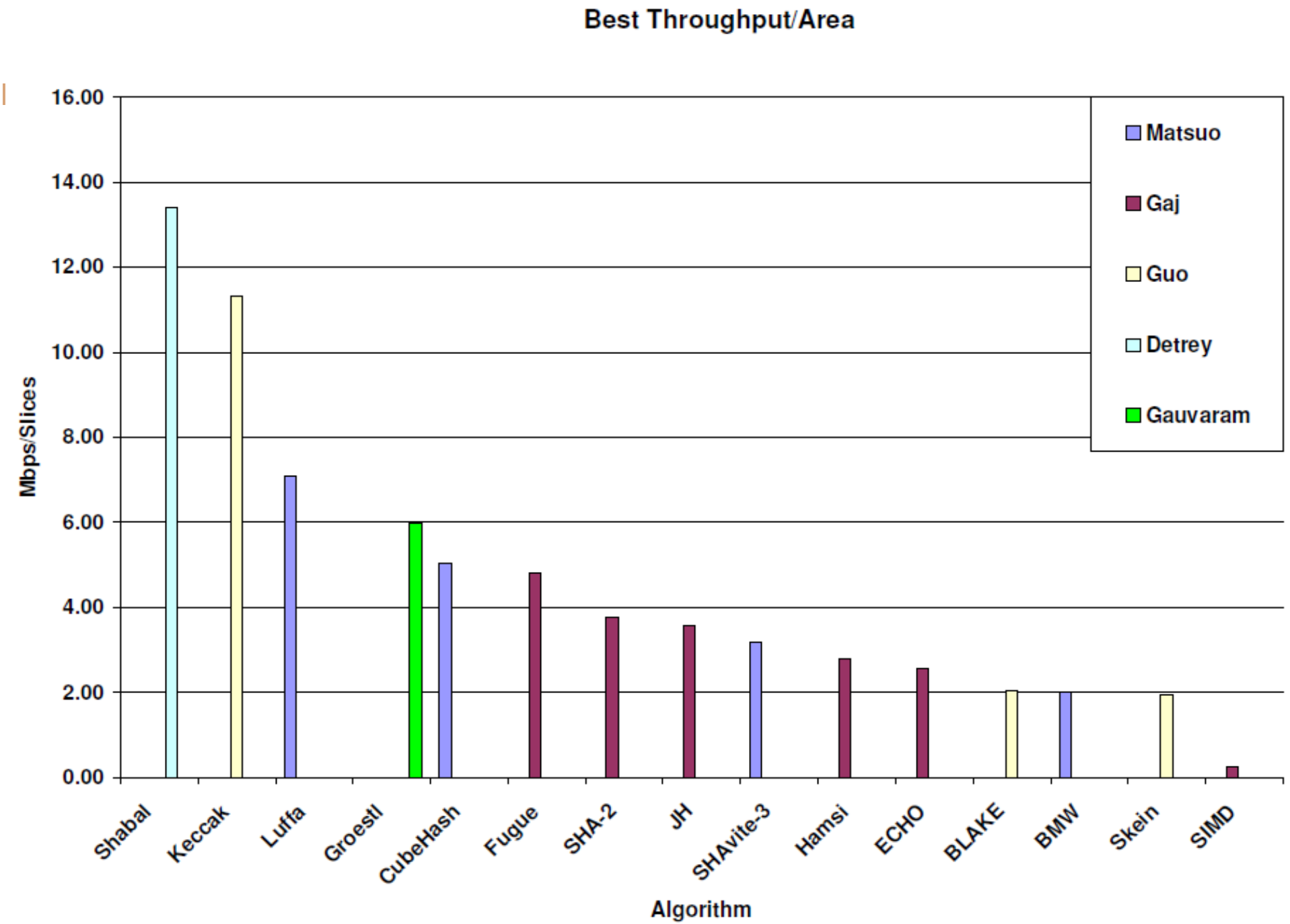
Function	Number of clock cycles per block: Baldwin	GMU	Differences Baldwin vs. GMU
Blake	40	21	horizontally folded by 4 vs. horizontally folded by 2
BMW	4	1	unbalanced architecture
ECHO	8	25	basic iterative vs. reuse of BIG.SubBytes
Fugue	7	2	unbalanced architecture
Groestl	10	21	parallel execution vs. quasi-pipelined architecture
Hamsi	6	3	2 vs. 1 clock cycle per round
SIMD	32	9	basic architecture vs. 4x unrolled architecture

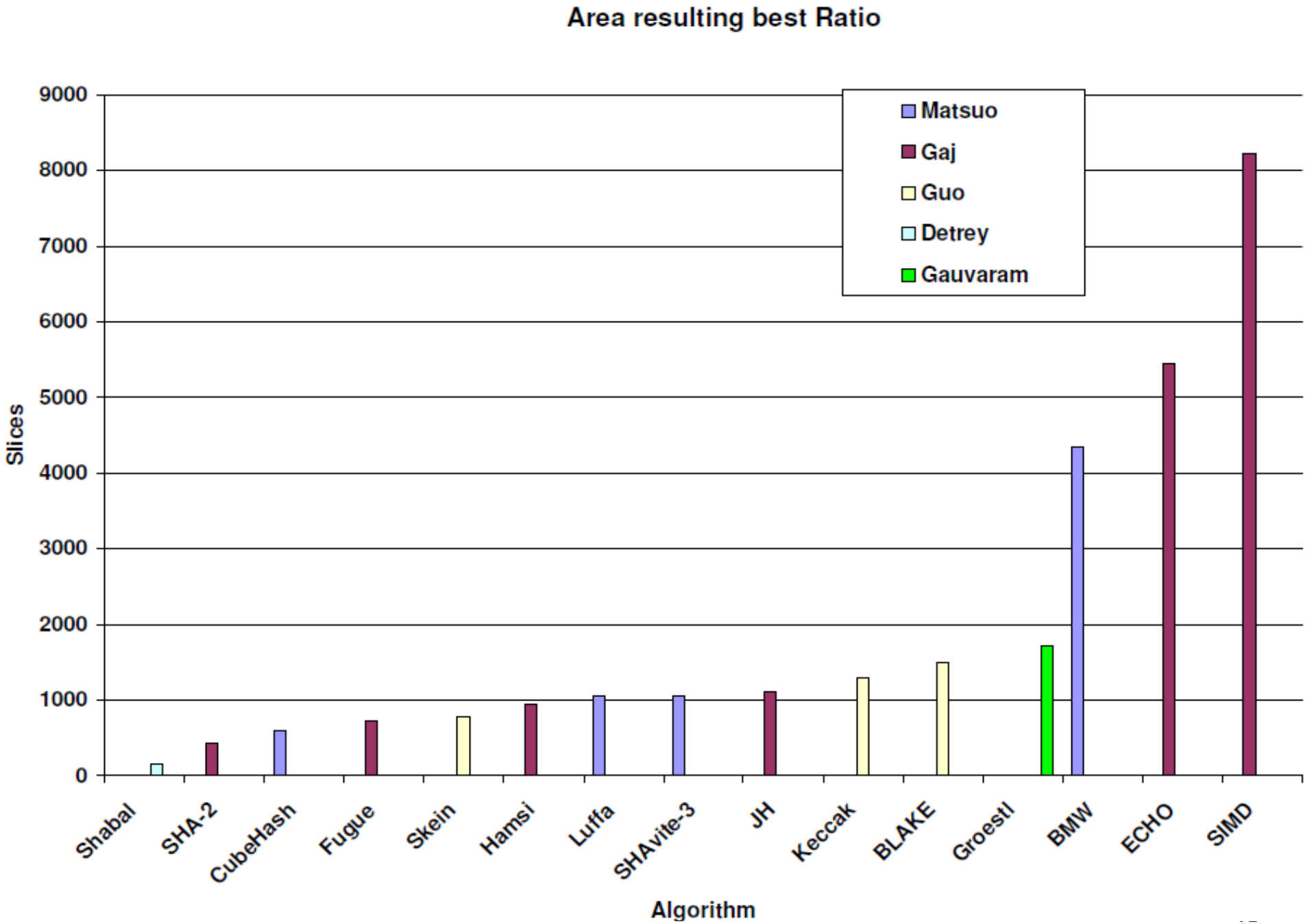
Explanation of Remaining Differences
Matsuo et al. vs. GMU

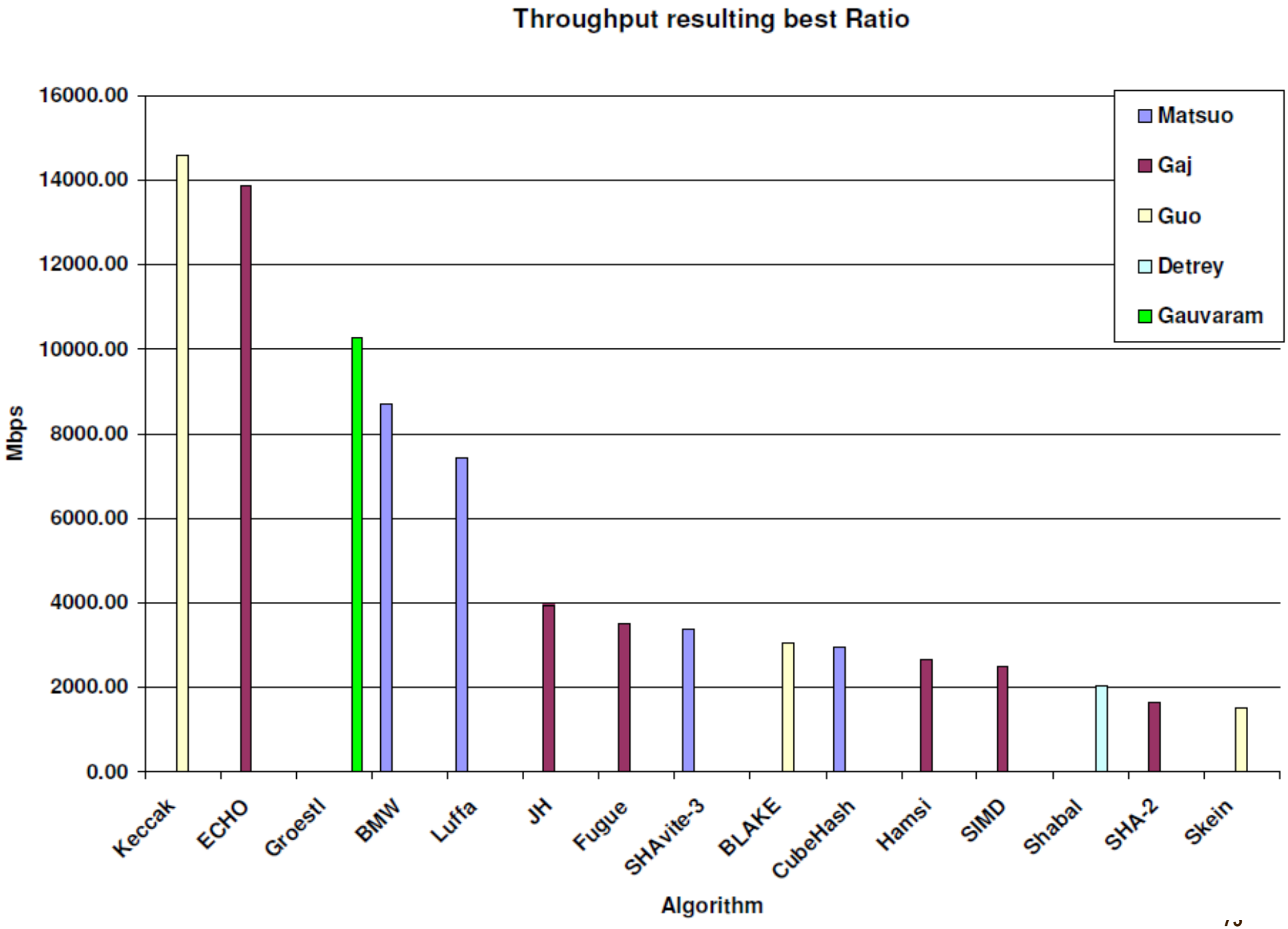
	Number of clock cycles per block		
Function	**Matsuo**	**GMU**	**Differences Matsuo vs. GMU**
ECHO	99	25	4x vertically folded architecture vs. 3/2 horizontally folded architecture
Groestl	10	21	parallel execution vs. quasi-pipelined architecture
SIMD	46	9	basic architecture vs. 4x unrolled architecture; message expansion and main rounds performed sequentially vs. in parallel

Keccak-256: 1024-bit block size vs. 1088-bit block size

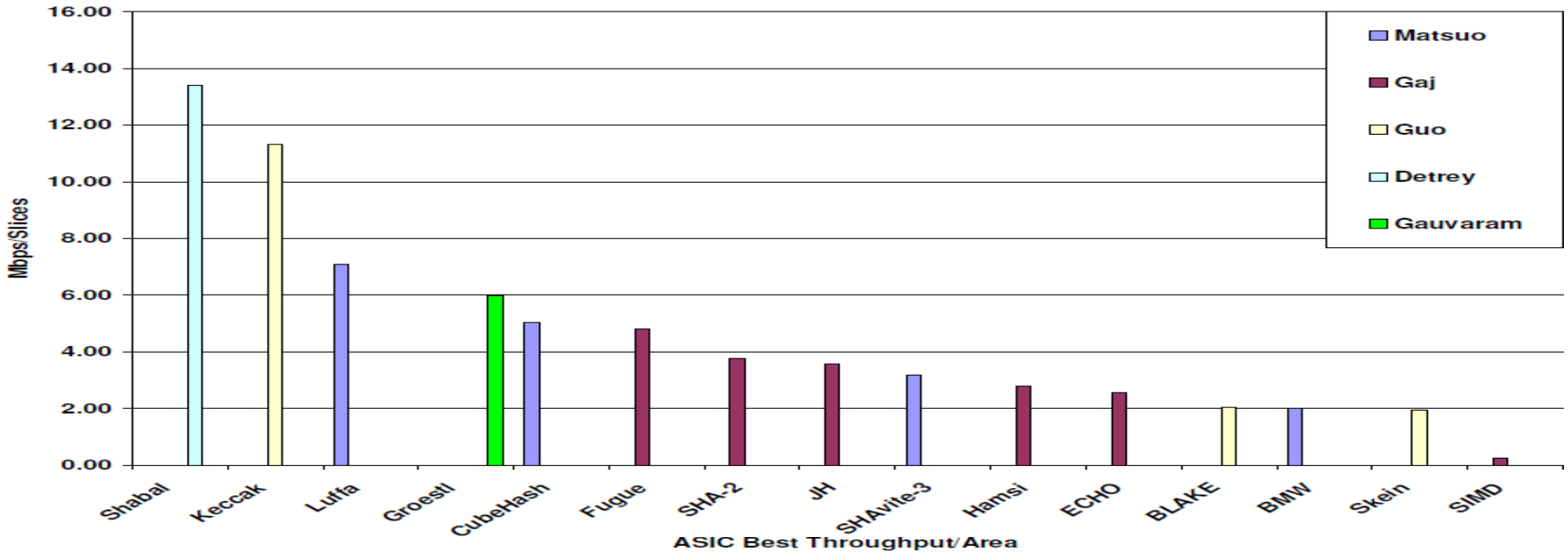
Skein-256-256 vs. Skein-512-256: 256 vs. 512-bit message block size and state size


Best Throughput/Area
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Mbps/Slices
Matsuo
Gaj
Guo
Detrey
Gauvaram
Shabal
Keccak
Luffa
Groestl
CubeHash
Fugue
SHA-2
JH
SHAvite-3
Hamsi
ECHO
BLAKE
BMW
Skein
SIMD
Algorithm

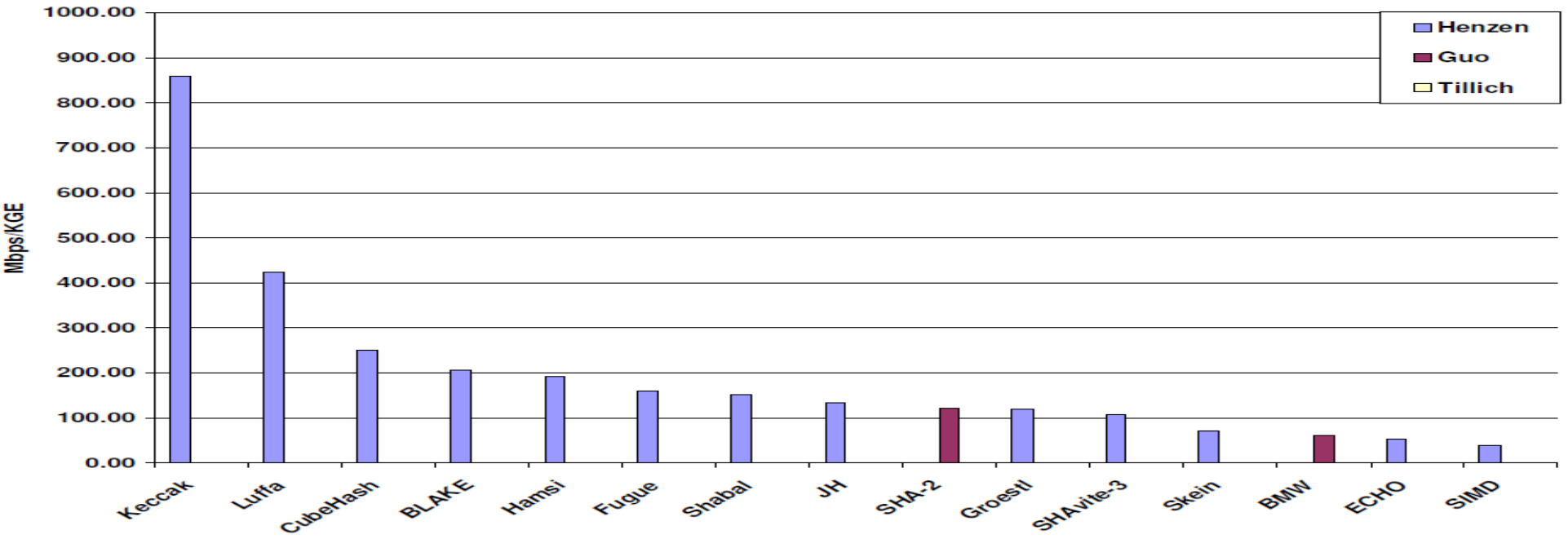

Area resulting best Ratio
Matsuo
Gaj
Guo
Detrey
Gauvaram
Slices
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Shabal
SHA-2
CubeHash
Fugue
Skein
Hamsi
Luffa
SHAvite-3
JH
Keccak
BLAKE
Groestl
BMW
ECHO
SIMD
Algorithm


Throughput resulting best Ratio
16000.00
14000.00
12000.00
10000.00
8000.00
6000.00
4000.00
2000.00
0.00
Mbps
Matsuo
Gaj
Guo
Detrey
Gauvaram
Keccak
ECHO
Groestl
BMW
Luffa
JH
Fugue
SHAvite-3
BLAKE
CubeHash
Hamsi
SIMD
Shabal
SHA-2
Skein
Algorithm

FPGA Best Throughput/Area



ASIC Best Throughput/Area



512-bit variant vs. 256-bit variant – Predicted Behavior

Group 1: Area: ⟷ Thr: ⟷ Thr/Area: ⟷

CubeHash, JH, Shabal, Skein

Group 2: Area: ↗ x2 Thr: ↗ x2 Thr/Area: ⟷

BMW, SIMD

Group 3: Area: ↗ Thr: ↗ Thr/Area: ↘

BLAKE, Groestl, SHAvite-3, SHA-2

Group 4: Area: ⟷ Thr: ↘ Thr/Area: ↘

ECHO, Keccak

Group 5: Area: ↗ Thr: ⟷ Thr/Area: ↘

Hamsi, Luffa

Group 6: Area: ↗ Thr: ↘ Thr/Area: ↘

Fugue

Hints for Designers of Hash Functions

- Easy way to predict **approximately** the change in speed and area when moving from a 256-bit to a 512-bit variant in **high-speed** hardware implementations

$$\frac{Area(512)}{Area(256)} \approx \frac{Datapath\_width(512)}{Datapath\_width(256)} = \frac{State\_size(512)}{State\_size(256)}$$

$$\frac{Thr(512)}{Thr(256)} \approx \frac{\frac{Block\_size(512)}{Block\_size(256)}}{\frac{Round\_no(512)}{Round\_no(256)}} = \frac{Block\_size\_ratio}{Round\_no\_ratio}$$

Pros and Cons of SHA-3 Candidates (1)

BLAKE

+ extremely flexible, multiple architectures
  obtained by horizontal, vertical, and mixed folding

BMW

+ good potential for pipelining
+ area efficient for high throughputs
- irregular structure
- difficulties with placing & routing
- quite complex folded architecture (late discovery),
  smaller but less efficient than the basic architecture
- need for an extra input/output clock

Pros and Cons of SHA-3 Candidates (2)

CubeHash

+ small area
+ good throughput to area ratio
+ very suitable for parallel processing
+ easy replacement for SHA-2 (similar in size and speed)
- relatively weak performance for short messages
- does not offer any significant performance advantage over SHA-2

ECHO

+ very flexible in terms of vertical folding
+ suitable for use of embedded block memories
  to implement AES S-boxes and/or T-boxes
+ good performance for short messages
- large area of the basic architecture

Pros and Cons of SHA-3 Candidates (3)

Fugue

+ very flexible in terms of vertical folding
+ suitable for use of embedded block memories
  to implement AES S-boxes and T-boxes
- relatively slow for very short messages
- area grows and throughput decreases for a 512-bit variant

Groestl

+ suitable for quasi-pipelining (pipelining with one message)
+ high throughput and throughput to area ratios
+ very flexible in terms of vertical folding
+ suitable for use of embedded block memories
  to implement AES S-boxes and T-boxes
- relatively large area of the basic architecture

Pros and Cons of SHA-3 Candidates (4)

Hamsi

+ suitable for use of embedded block memories
  to implement message expansion
- limited flexibility, no known folded architectures

JH

+/- good potential for folding but with limited area improvement

Keccak

+ very high throughput and throughput to area ratio
  especially for a 256-bit variant
+ good potential for pipelining
- limited flexibility, no known folded architectures

Pros and Cons of SHA-3 Candidates (5)

Luffa

+ very high throughput and throughput to area ratio for both 256 and 512-bit variant
+ good flexibility: straightforward folded architectures for medium-speed implementations

Shabal

+ extremely small area and high throughput to area ratio for Xilinx FPGAs (does not carry to Altera FPGAs or ASICs)
+ very suitable for parallel processing
- relatively small throughput of the basic architecture
- relatively weak performance for short messages

Pros and Cons of SHA-3 Candidates (6)

SHAvite-3

+ flexible in terms of vertical folding
+ suitable for use of embedded block memories
  to implement AES S-boxes and T-boxes
- complex key scheduling, difficult to fold or unroll

SIMD

- big area of the basic architecture
- by far the worst throughput to area ratio
- most time consuming to implement and debug
- complex message expansion unit
+ good potential for folding

Pros and Cons of SHA-3 Candidates (7)

Skein

+ good potential for pipelining

- relatively small throughput of the basic architecture before pipelining

More About our Designs & Tools

- CHES 2010 paper
  - **Methodology**
  - Results for 256-bit variants

  FPL 2010 paper
  - ATHENa features
  - Case studies

- Cryptology e-Print Archive, 2010/445, last updated on Oct. 10, 2010
  - **Detailed hierarchical block diagrams,**
    - **60 diagrams for 15 functions**
  - **Corresponding formulas for execution time and throughput**

- ATHENa web site
  - **Most recent results**
  - Comparisons with results from other groups
  - **Optimum options of tools**

Thank you!

Questions?

Questions?

CERG: **http:/cryptography.gmu.edu**

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