

Performance Analysis of the SHA-3 Candidates on Exotic Multi-Core Architectures

Joppe W. Bos¹ Deian Stefan²

¹Laboratory for Cryptologic Algorithms
EPFL, Station 14, CH-1015 Lausanne, Switzerland

²Department of Electrical Engineering
The Cooper Union, NY 10003, New York, USA



Motivation

SHA-3 competition

Nov '08 Dec '08 July '09 3Q '10 2Q 2012
64 → 51 → 14 → ≈ 5 → ≈ 1

Abacus, ARIRANG, AURORA, **BLAKE**, Blender, **BMW**, BOOLE, Cheetah, CHI, CRUNCH, **CubeHash**, DCH, Dynamic SHA, Dynamic SHA2, **ECHO**, ECOH, EDON-R, EnRUPT, ESSENCE, FSB, **Fugue**, **Grøstl**, **Hamsi**, **JH**, **Keccak**, Khichidi-1, LANE, Lesamnta, **Luffa**, LUX, MCSSHA-3, MD6, MeshHash, NaSHA, SANDstorm, Sarmal, Sgàil, **Shabal**, SHAMATA, **SHAvite-3**, **SIMD**, **Skein**, Spectral Hash, StreamHash, SWIFFTX, Tangle, TIB3, Twister, Vortex, WaMM, Waterfall

Main evaluation criteria

- Security
- Cost (computational efficiency, memory requirements)
- Algorithmic and implementation characteristics

NIST states that is it preferable if the algorithm
“can be implemented securely and efficiently on a wide variety of platforms.”

eBASH results (accessed 2 August 2010)

BMW	6.4	Shabal	10.2
Skein	6.6	BMW	11.5
Shabal	8.2	SHA-256	19.6
BLAKE	10.2	BLAKE	20.9
SIMD	11.5	Luffa	32.8
Keccak	12.4	ECHO	34.5
ch-16-32	13.2	SHAvite-3	34.7
Luffa	13.5	Keccak	38.3
SHA-256	15.4	Fugue	41.6
JH	17.0	Grøstl	42.6
Grøstl	21.3	Skein	52.1
Fugue	22.8	ch-16-32	72.4
ECHO	24.4	SIMD	96.1
SHAvite-3	27.4	JH	127.7
Hamsi	30.5		

eBASH results (accessed 2 August 2010)

amd64 Intel Xeon (E7310) MMX, SSE SSE2, SSE3 SSSE3	{	BMW	6.4	Shabal	10.2	x86 AMD Athlon (622) MMX 3DNow!
		Skein	6.6	BMW	11.5	
		Shabal	8.2	SHA-256	19.6	
		BLAKE	10.2	BLAKE	20.9	
		SIMD	11.5	Luffa	32.8	
		Keccak	12.4	ECHO	34.5	
		ch-16-32	13.2	SHAvite-3	34.7	
		Luffa	13.5	Keccak	38.3	
		SHA-256	15.4	Fugue	41.6	
		JH	17.0	Grøstl	42.6	
		Grøstl	21.3	Skein	52.1	
		Fugue	22.8	ch-16-32	72.4	
		ECHO	24.4	SIMD	96.1	
		SHAvite-3	27.4	JH	127.7	
		Hamsi	30.5			

eBASH results (accessed 2 August 2010)

amd64
Intel Xeon
(E7310)
MMX, SSE
SSE2, SSE3
SSSE3

BMW	6.4	Shabal	10.2
Skein	6.6	BMW	11.5
Shabal	8.2	SHA-256	19.6
BLAKE	10.2	BLAKE	20.9
SIMD	11.5	Luffa	32.8
Keccak	12.4	ECHO	34.5
ch-16-32	13.2	SHAvite-3	34.7
Luffa	13.5	Keccak	38.3
SHA-256	15.4	Fugue	41.6
JH	17.0	Grøstl	42.6
Grøstl	21.3	Skein	52.1
Fugue	22.8	ch-16-32	72.4
ECHO	24.4	SIMD	96.1
SHAvite-3	27.4	JH	127.7
Hamsi	30.5		

x86
AMD Athlon
(622)
MMX
3DNow!

Evaluate performance on two *exotic* platforms

Cell Broadband Engine architecture (Cell)

NVIDIA Graphics Processing Units (GPUs)

- Performance analysis framework of all SHA-3 candidates
- Specific optimization techniques for each of our target platforms
- Implementations of all non-AES based candidates

Multi-stream fixed-length message processing applications:

- Cryptanalytic applications (e.g., cube testers, differential attacks),
- Optimal asymmetric encryption (ANSI, IEEE, PKCS)
mask generation function based on a fixed-input size hash-function
- Hash-based Message Authentication Codes (HMAC)

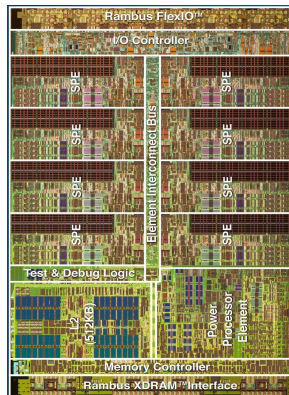
Exotic platform I: Cell architecture

8 Synergistic Processing Elements (SPEs)

1 Dual-threaded Power Processing Element

The SPEs contain

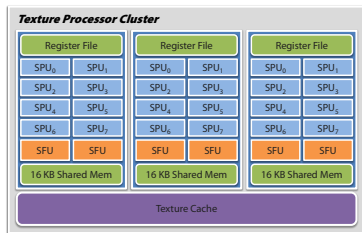
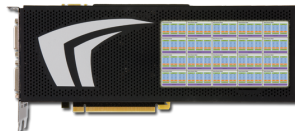
- a Synergistic Processing Unit (SPU)
 - Access to 128 registers of 128-bit
 - SIMD operations
 - Dual pipeline (odd and even)
 - Rich instruction set
 - all distinct binary operations $f : \{0, 1\}^2 \rightarrow \{0, 1\}$ are available
 - In-order processor
- 256 KB of fast local memory (Local Store)
- Memory Flow Controller (MFC)



Available in BladeServer, PlayStation 3, PCIe card

Exotic Platform II: NVIDIA GT200 GPUs

- A TPC contains
 - 3 Simultaneous Multiprocessors (SMs)
 - 24KB texture cache
- The SMs contain
 - 8 Streaming Processors (SPs): support simple 32-bit operations
 - 16384 32-bit registers
 - 16KB 16-way banked shared memory
 - 8KB constant memory cache
 - Texture cache read port
 - 2 special function units (SFUs)
 - Multithreaded scheduler
- Each GTX200 GPU contains 8-10 TPCs
 - total of 192-240 SPs/GTX200
 - GeForce GTX 295 graphics card: 480 SPs



Counting instructions

Estimating performance by counting 32-bit instructions

Is this approach the holy grail to predict benchmarks?

Counting instructions

Estimating performance by counting 32-bit instructions

Is this approach the holy grail to predict benchmarks?

Of course not!



Pros	Cons
Give a <u>crude</u> performance estimate on a hypothetical 32-bit architecture	Ignores all overhead related to I/O (loading/storing, cache misses etc)
Use as a base for more refined architecture-specific estimates	Ignores flexibility provided by SIMD-coprocessors
Not biased to any candidate	State size not considered

Very suitable for Cell and GPU

Rich 32-bit instruction set, no SIMD-coprocessors

AES-inspired candidates

Build the compression function using

- AES-round function $\rightarrow$ ECHO, SHAvite-3
- AES-like (byte-oriented) operations $\rightarrow$ Fugue, Grøstl

AES-inspired candidates

Build the compression function using

- AES-round function → ECHO, SHAvite-3
- AES-like (byte-oriented) operations → Fugue, Grøstl

Motivation

- Security: no known attacks on AES-128
- Speed: AES (and its components) is fast on many platforms
- Cost: Intel AES instruction set → simple, efficient, secure

AES-inspired candidates

Build the compression function using

- AES-round function $\rightarrow$ ECHO, SHAvite-3
- AES-like (byte-oriented) operations $\rightarrow$ Fugue, Grøstl

Motivation

- Security: no known attacks on AES-128
- Speed: AES (and its components) is fast on many platforms
- Cost: Intel AES instruction set $\rightarrow$ simple, efficient, secure

Function	b	AES (R)	SB	MC4	MC8	MC16	xor(byte)
AES-128	16	10	-	-	-	-	16
ECHO-256	192	256	-	512	-	-	448
Fugue-256	4	-	32	-	-	2	60
Grøstl-256	64	-	1280	-	160	-	1472
SHAvite-3-256	64	52	-	-	-	-	1280

Performance Considerations

MixColumns (MC) operation has the greatest effect on performance

- MCX: each column of state is multiplied by a $X \times X$ matrix
- Typically implemented using XTIME function and several xors

XTIME (byte-value shift + c_{xor}) and xor operations

- For MCX ($X \in \{8, 16\}$) compute the double and quadruple
- All constants in Fugue < 8 $\therefore$ no need for octuple in MC16
 - $2 \cdot X$ XTIME operations, and $\#xor$ depends on the constants
 - Exploit circulant matrix structure

“T-table” approach

Combine the MC and substitution step into table-lookup

Performance Considerations

MixColumns (MC) operation has the greatest effect on performance

- MCX: each column of state is multiplied by a $X \times X$ matrix
- Typically implemented using XTIME function and several xors

MCX	XTIME	xor (byte)	T-table, operate on X bytes		
			size of table(s) in bytes	xor	rotate
MC4 (AES)	4	16	1,024	3	3
			4,096	3	0
MC8 (Grøstl)	16	104	2,048	7	7
			16,384	7	0
MC16 (Fugue)	32	148	4,096	15	15
			65,536	15	0

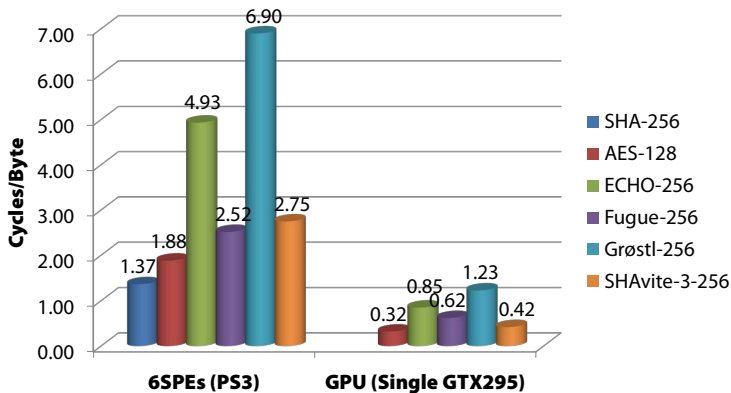
Note: These estimates are an upper bound!

Example of a *better* implementation:

[Osvik,Bos,Stefan,Canright-FSE10]: AES MC4: 3 XTIME, 15 xor

Estimated Results

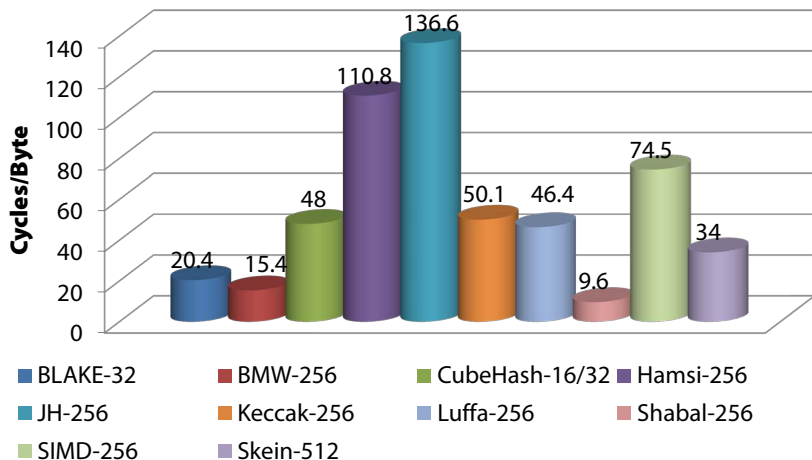
Use the AES results from [OBSC-10] to estimate performance



Counting operations, non-AES based candidates

Hash function	<i>b</i>	add	sub <i>csub</i>	mul	and	nand <i>andc</i>	eqv	or <i>orc</i>	rotate	shift	xor	Cycles / byte
Hash functions operating on 32-bit words												
BLAKE-32	64	480	-	-	-	-	-	-	320	-	508	20.4
BMW-256	64	296	58	-	-	-	-	-	212	144	277	15.4
CubeHash-16/32	32	512	-	-	-	-	-	-	512	-	512	48.0
Hamsi-256	4	-	-	-	24	12	-	24	72	24	287	110.8
JH-256	64	-	-	-	1792	1152	288	688	-	800	4024	136.6
Keccak-256	136	-	-	-	684	96	144	480 144	1248	204	3810	50.1
Luffa-256	32	-	-	-	144	-	96	96	392	-	756	46.4
Shabal-256	64	52	16	96	-	48	48	-	112	-	242	9.6
SIMD-256	64	817	901 256	419	852	-	-	256	288	804	176	74.5
Hash functions operating on 64-bit words												
Skein-512	64	497	-	-	1	-	-	-	288	-	305	17.0

Non-AES candidates' estimated performance



Reality check

Estimate by counting instructions	{	Shabal	9.6	Shabal	10.2	{	x86 AMD Athlon (622) MMX 3DNow!
		BMW	15.4	BMW	11.5		
		BLAKE	20.4	BLAKE	20.9		
		Luffa	46.4	Luffa	32.8		
		ch-16-32	48.0	Keccak	38.3		
		Keccak	50.1	ch-16-32	72.4		
		SIMD	74.5	SIMD	96.1		
		JH	136.6	JH	127.7		

Reality check

Estimates are too *optimistic*:

no rotate instructions, ignores table lookups

Estimates are too *pessimistic*:

AMD Athlon can sustain a throughput of 3 instructions per cycle

Estimate by counting instructions	{	Shabal	9.6	Shabal	10.2	{	x86 AMD Athlon (622) MMX 3DNow!
		BMW	15.4	BMW	11.5		
		BLAKE	20.4	BLAKE	20.9		
		Luffa	46.4	Luffa	32.8		
		ch-16-32	48.0	Keccak	38.3		
		Keccak	50.1	ch-16-32	72.4		
		SIMD	74.5	SIMD	96.1		
		JH	136.6	JH	127.7		

Improve estimates: consider *all* characteristics of the target platform

Results: hashing 25KB messages

Algorithm	SPE		NVIDIA GTX 295 GPU	
	Cycles per byte	Throughput (Gb/sec)	Cycles per byte	Throughput (Gb/sec)
SHA-256 [BCO08]	8.2	3.1	-	-
BLAKE-32	5.0 (4.5)	5.1 (5.7)	0.27 [0.13] (0.13)	36.8 (76.4)
BMW-256	4.2 (3.7)	6.2 (6.9)	0.27 [0.27] (0.10)	36.8 (99.4)
CubeHash-16/32	11.6 (9.9)	2.2 (2.6)	0.36 [0.35] (0.34)	27.6 (29.2)
Hamsi-256	32.2 (26.9)	0.8 (1.0)	5.19 [0.66] (0.64)	1.91 (15.5)
JH-256	31.5 (29.8)	0.8 (0.9)	0.76 [0.75] (0.67)	13.1 (14.8)
Keccak-256	13.0 (11.1)	2.0 (2.3)	0.56 [0.56] (0.31)	17.7 (32.1)
Luffa-256	11.5 (10.1)	2.2 (2.5)	0.35 [0.34] (0.32)	28.4 (31.1)
Shabal-256	3.5 (2.8)	7.2 (9.2)	0.69 [0.56] (0.07)	14.4 (141.9)
SIMD-256	22.6 (19.0)	1.1 (1.4)	3.60 [3.60] (0.43)	2.76 (23.1)
Skein-512	13.7 (12.1)	1.9 (2.1)	0.46 [0.29] (0.22)	22.1 (45.2)

Results: hashing 25KB messages

Algorithm	SPE		NVIDIA GTX 295 GPU	
	Cycles per byte	Throughput (Gb/sec)	Cycles per byte	Throughput (Gb/sec)
SHA-256 [BCO08]	8.2	3.1	-	-
BLAKE-32	5.0 (4.5)	5.1 (5.7)	0.27 [0.13] (0.13)	36.8 (76.4)
BMW-256	4.2 (3.7)	6.2 (6.9)	0.27 [0.27] (0.10)	36.8 (99.4)
CubeHash-16/32	11.6 (9.9)	2.2 (2.6)	0.36 [0.35] (0.34)	27.6 (29.2)
Hamsi-256	32.2 (26.9)	0.8 (1.0)	5.19 [0.66] (0.64)	1.91 (15.5)
JH-256	31.5 (29.8)	0.8 (0.9)	0.76 [0.75] (0.67)	13.1 (14.8)
Keccak-256	13.0 (11.1)	2.0 (2.3)	0.56 [0.56] (0.31)	17.7 (32.1)
Luffa-256	11.5 (10.1)	2.2 (2.5)	0.35 [0.34] (0.32)	28.4 (31.1)
Shabal-256	3.5 (2.8)	7.2 (9.2)	0.69 [0.56] (0.07)	14.4 (141.9)
SIMD-256	22.6 (19.0)	1.1 (1.4)	3.60 [3.60] (0.43)	2.76 (23.1)
Skein-512	13.7 (12.1)	1.9 (2.1)	0.46 [0.29] (0.22)	22.1 (45.2)

4 input streams per SPE

6 SPEs in a PS3, 16 SPEs in a BladeServer QS{20,21,22}

Estimates and benchmark results are consistent

Results: hashing 25KB messages

Algorithm	SPE		NVIDIA GTX 295 GPU	
	Cycles per byte	Throughput (Gb/sec)	Cycles per byte	Throughput (Gb/sec)
SHA-256 [BCO08]	8.2	3.1	-	-
BLAKE-32	5.0 (4.5)	5.1 (5.7)	0.27 [0.13] (0.13)	36.8 (76.4)
BMW-256	4.2 (3.7)	6.2 (6.9)	0.27 [0.27] (0.10)	36.8 (99.4)
CubeHash-16/32	11.6 (9.9)	2.2 (2.6)	0.36 [0.35] (0.34)	27.6 (29.2)
Hamsi-256	32.2 (26.9)	0.8 (1.0)	5.19 [0.66] (0.64)	1.91 (15.5)
JH-256	31.5 (29.8)	0.8 (0.9)	0.76 [0.75] (0.67)	13.1 (14.8)
Keccak-256	13.0 (11.1)	2.0 (2.3)	0.56 [0.56] (0.31)	17.7 (32.1)
Luffa-256	11.5 (10.1)	2.2 (2.5)	0.35 [0.34] (0.32)	28.4 (31.1)
Shabal-256	3.5 (2.8)	7.2 (9.2)	0.69 [0.56] (0.07)	14.4 (141.9)
SIMD-256	22.6 (19.0)	1.1 (1.4)	3.60 [0.65] (0.43)	2.76 (23.1)
Skein-512	13.7 (12.1)	1.9 (2.1)	0.46 [0.29] (0.22)	22.1 (45.2)

680 blocks of 64 threads (43520 streams) per GPU
in [brackets] the compression function only
in (parentheses) the estimates

GPU estimated vs. observed results

Hamsi

- Message expansion requires random reads from **huge 32KB table**
- Atomic xor to global memory, random access reads, use of 2 kernels, and large #threads/stream lead to poor performance
- Note: Compression function performance closely agrees with estimate

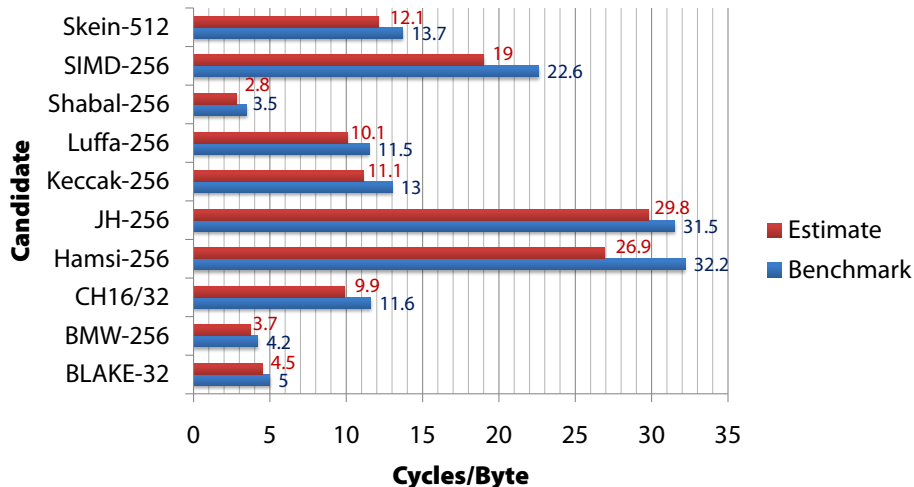
SIMD

- Memory pressure: message expansion 4Kb in addition to internal state
- Use of **slow local memory** required in multi-stream implementation

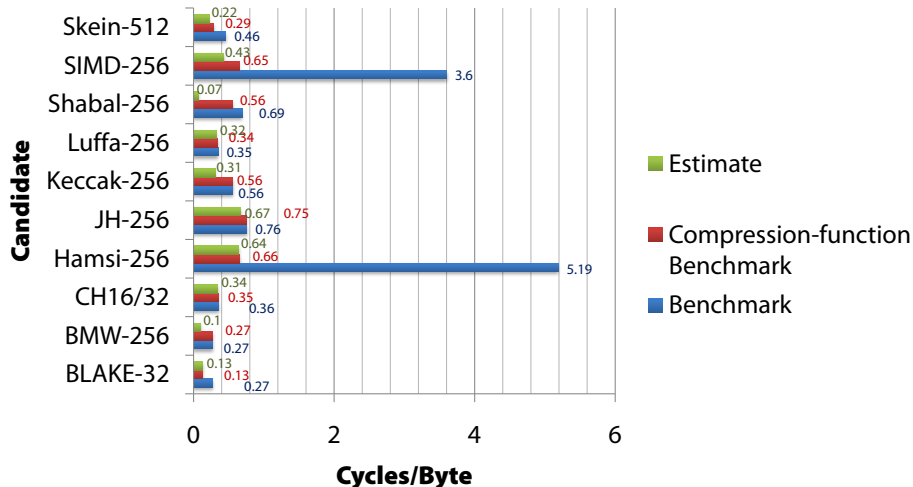
Shabal

- Addressing **compiler bug** → cannot inline function in permutations
- Code will be re-benchmarked upon new release of CUDA

SPE result-visualization



GPU results-visualization



Conclusions

- Throughput analysis of *all* 2nd-round SHA-3 candidates
→ useful as base for architecture-specific estimates
- Estimated performance of *all* 2nd-round SHA-3 candidates on Cell and GPU → target-specific optimizations considered in the estimates
- Implemented *all* non-AES based 2nd-round SHA-3 candidates
→ benchmarked results agree with most estimates

We hope this work can assist in the decision process of the SHA-3 competition