Lightweight Implementations of SHA-3 Candidates on FPGAs

Jens-Peter Kaps   Panasayya Yalla   Kishore Kumar Surapathi
Bilal Habib   Susheel Vadlamudi   **Smriti Gurung**
John Pham

Cryptographic Engineering Research Group (CERG)
http://cryptography.gmu.edu
Department of ECE, Volgenau School of Engineering,
George Mason University, Fairfax, VA, USA

12th International Conference on Cryptology in India
Indocrypt 2011

Outline

Hash Function Competition

- A hash algorithm reads an arbitrary length message and produces a fixed bit string called hash value/message digest.
- Main applications: Digital signatures, Message Authentication Codes (MAC), Universal Unique IDentifier(UUID/GUID), password tables and many more.
- NIST competition for new secure hash algorithm SHA-3
  - Announced in Nov 2007, 64 entries submitted.
  - 14 selected for Round 2.
  - Currently in Round 3 $\rightarrow$ 5 finalists.
- NIST's selection criteria: Security, HW/SW speed, scalability.

Hash Function Competition

- A hash algorithm reads an arbitrary length message and produces a fixed bit string called hash value/message digest.
- Main applications: Digital signatures, Message Authentication Codes (MAC), Universal Unique IDentifier(UUID/GUID), password tables and many more.
- NIST competition for new secure hash algorithm SHA-3
  - Announced in Nov 2007, 64 entries submitted.
  - 14 selected for Round 2.
  - Currently in Round 3 $\rightarrow$ 5 finalists.
- NIST's selection criteria: Security, HW/SW speed, scalability.

Motivation

- Analyze performance of candidates in a constrained FPGA environment $\Rightarrow$ determine scalability on FPGAs.

Previous Work on SHA-3 Candidates

- Several Throughput/Area optimized implementations on FPGAs were published: Gaj et al.[CHES 2010], Matsuo et al.[SHA-3 conference 2010], Baldwin et al.[SHA-3 conference 2010].
- Only two specific for low-area implementations of SHA-3 finalists: Kerckhof et al.[HASH 2011], Jungk et al.[Reconfig 2011].

Previous Work on SHA-3 Candidates

- Several Throughput/Area optimized implementations on FPGAs were published: Gaj et al.[CHES 2010], Matsuo et al.[SHA-3 conference 2010], Baldwin et al.[SHA-3 conference 2010].
- Only two specific for low-area implementations of SHA-3 finalists: Kerckhof et al.[HASH 2011], Jungk et al.[Reconfig 2011].

Problem: Rating algorithm performance when

- Implementations are on different devices,
- made with different implementation goals and features,
- vary in both: area and throughput, and
- support different I/O interface widths.

Our Goal:

Comprehensive set of lightweight implementations of all Round 2 SHA-3 Candidates (except SIMD) and all SHA-3 Finalists.

- All optimized for the same target → maximum Throughput to Area ratio for given area budget.
- All use the same standardized interface.
- Implemented on different families for fair comparison with other reported results.

Target Details:

- Xilinx Spartan 3, low cost FPGA family
- Budget: 400-600 slices, 1 Block RAM (BRAM)
- Implemented 256 bit digest versions only

Assumptions

- Implementing for minimum area alone can lead to unrealistic run-times.
- $\Rightarrow$ Target: Achieve the maximum Throughput/Area ratio for a given area budget.
- Realistic scenario:
  - System on Chip: Certain area only available.
  - Standalone: Smaller Chip, lower cost, but limit to smallest chip available, e.g. 768 slices on smallest Spartan 3 FPGA.
- Makes fair comparison of lightweight implementations possible.

Interface and Protocol

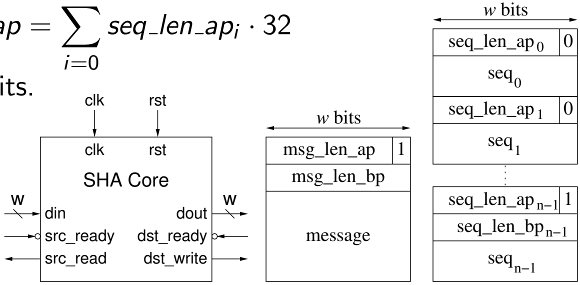
Based on Interface and I/O Protocol from Gaj et al.[CHES 2010].

- msg_len_ap, seq_len_ap (after padding) in 32-bit words.
- msg_len_bp, seq_len_bp (before padding) in bits.

$$msg\_len\_bp = \sum_{i=0}^{n-2} seq\_len\_ap_i \cdot 32 + seq\_len\_bp_{n-1}$$

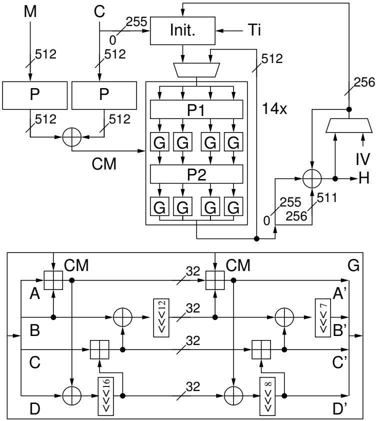
$$msg\_len\_ap = \sum_{i=0}^{n-1} seq\_len\_ap_i \cdot 32$$

$w = 16$ bits.



a)SHA Interface b)SHA Protocol

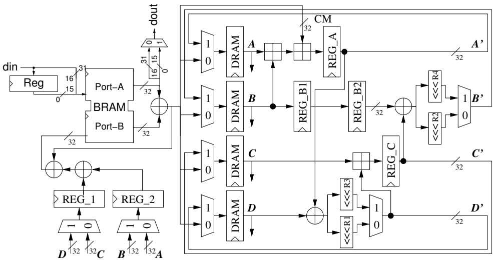
BLAKE-256 Algorithm



Key Features

- Salt value: A user Dependant constant 128 bits set all to 0
- 8 G functions : XOR, addition, shifting.
- P1,P2 : Permutation
- Blake scales very well.
- Folded up to 4 times vertically and 4 times horizontally.

BLAKE-256 Implementation

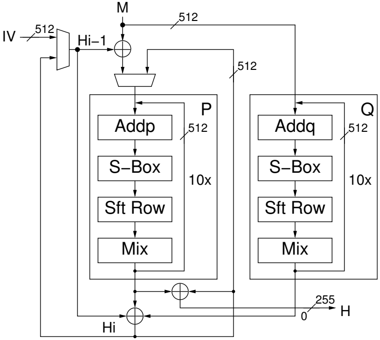


Implementation

- Salt : BRAM
- State: DRAM
- Quasi pipelined Half G function
- Registers: Reduce critical path

- Permutation causes a large controller with 210 addresses.
- BRAM contains constants, message, IV, intermediate hash.
- **Scalability:** Unfolding leads to worse TP/A.
- **Improvement:** Rescheduling of G results in 290 clock per block versus 350 .

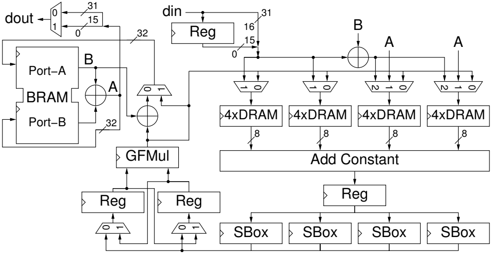
Grøstl Algorithm



Key Features

- Based on AES like architecture
- S-BOX, shift rows, Mixed columns
- Grøstl scales well, like AES.
- Folded up to 8 times vertically.
- Small storage requirements.
- Uses many narrow memory accesses in parallel (8 per column).

Grøstl Implementation

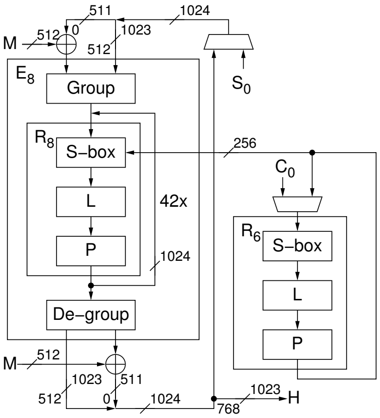


Implementation

- State p,q : DRAM
- Shift Rows : how data accessed from DRAM
- Mix Column : GF-multiplier(half multiplier)

- Finalization takes as many clock cycles as 1 block.
- BRAM stores only intermediate hash and IV.
- One new column every 3 clock cycles, P & Q interleaved.
- **Scalability:** Reducing number of clock cycles per column by adding S-Boxes and/or GF-Multiplier.

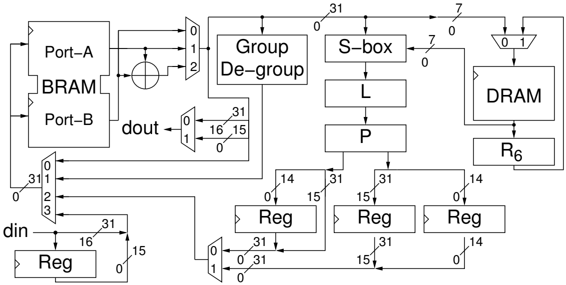
JH Algorithm



Key Features

- Grouping: reordering of 1024 bits state
- SBOX : Permutation
- Linear transformation : rotation and XOR
- De-grouping: inverse of grouping
- Permutation , grouping, and de-grouping makes scaling difficult
- Folding increases size

JH Implementation

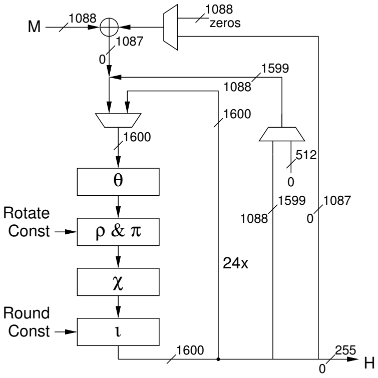


Implementation

- State: BRAM
- R8 function: Implemening 8X2 S-BOX for R8(S0 and S1)
- R6 function :2 S-BOX for R6(S0)

- 32-bit datapath to maximize use of BlockRAM.
- On-the-fly generation of round constants.
- **Scalability:** 64-bit datapath only viable without BlockRAM.
- **Improvement:** Group can be performed on M and de-group only on H Kerckhof et al.[ECRYPT II 2011].

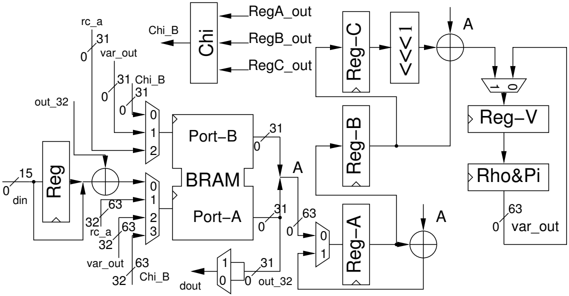
Keccak Algorithm



Key Features

- θ simple XOR
- ρ, π rotation and reordering operate on columns
- χ logical operation on rows
- Dependency on Previous states prevents folding.

Keccak Implementation

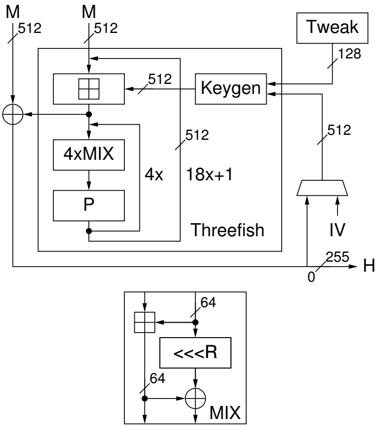


Implementation

- Round constants, States: BRAM
- Quasi-pipelined θ & ρ & π

- Fixed rotations turn into variable rotator for small datapaths.
- $\boxed{\rho \ \& \ \pi}$ contains the rotator.
- **Scalability:** 64-bit datapath only viable without BlockRAM.
- Adding 2 more 64-bit registers saves approx 700 clock cycles.

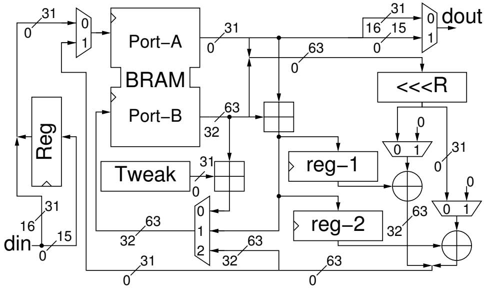
Skein Algorithm



Key Features

- Mix function : Addition, Rotation and XOR
- Tweak constant : Key Generation for each block
- 64-bit adders lead to long delay.
- Algorithm cannot be folded.

Skein Implementation

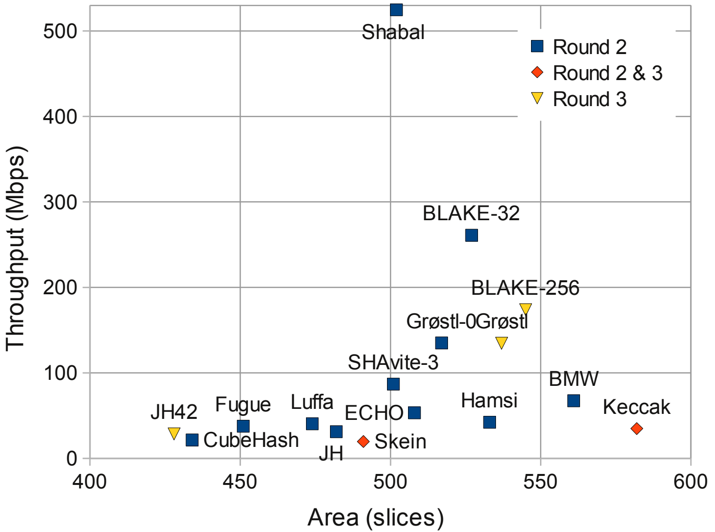


Implementation

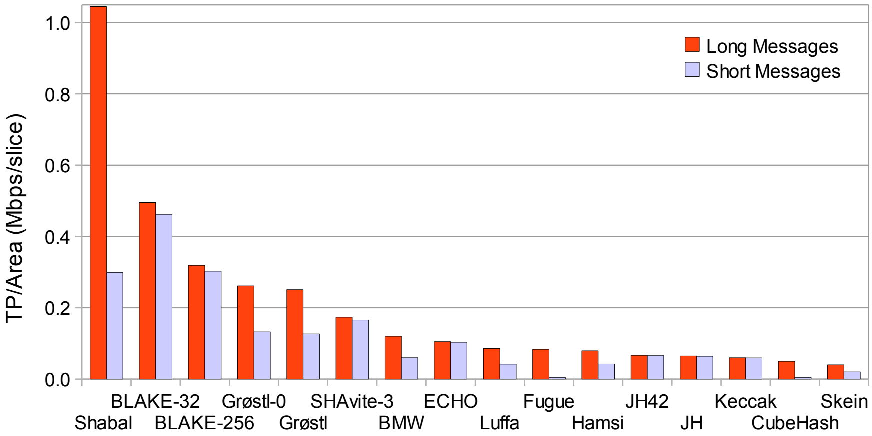
- State: BRAM
- Key Generation,Mix : 32 bit adder
- 32 bit adder leads critical path through barrel shifter.

- Barrel shifter is single largest block in the design (192 slices).
- Finalization takes as many clock cycles as 1 block hash.
- **Scalability:** Running Keygen and MIX in parallel.
- **Improvement:** Addition of Registers to cut down the critical path delay.

Throughput versus Area on Spartan-3



Ranking by Throughput over Area on Spartan-3



Algorithms with finalization rounds perform worse for short messages.

Implementation Summary of Finalists for Long Messages

Algorithm	Block Size (bits) b	Clock Cycles to hash N blocks $clk = st + (l + p) \cdot N + end$	Throughput $\frac{b}{(l+p) \cdot T}$
BLAKE-256	512	$2 + (32 + 318) \cdot N + 17$	$512/(350 \cdot T)$
Grøstl	512	$2 + (32 + 515) \cdot N + 532$	$512/(547 \cdot T)$
JH42	512	$35 + (32 + 1813) \cdot N - 15$	$512/(1845 \cdot T)$
Keccak	1088	$2 + (68 + 3696) \cdot N + 17$	$1088/(3764 \cdot T)$
Skein	512	$5 + (32 + 2407) \cdot N + 2423$	$512/(2439 \cdot T)$

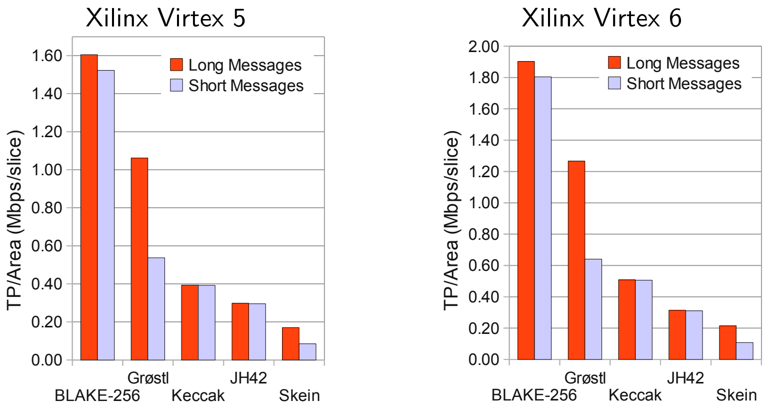
- st: Clock cycles computing initial steps before processing.
- $l + p$: Loading and processing cycles per block.
- end : Clock cycles needed for finalization and output of hash.
- st and end ignored for long messages as their influence goes toward zero.

Implementation Results of Finalists on Xilinx Spartan-3

Algorithm	Area (slices)	Block RAMs	Maximum Delay (ns) T	Long Message Throughput (Mbps)	Long Message TP/Area (Mbps/slice)	Short Message Throughput (Mbps)	Short Message TP/Area (Mbps/slice)
BLAKE-256	545	1	8.42	173.8	0.32	164.8	0.302
Grøstl	537	1	6.95	134.6	0.25	68.1	0.127
JH42	428	1	9.74	28.5	0.07	28.2	0.066
Keccak	582	1	8.30	34.8	0.06	34.7	0.060
Skein	491	1	10.68	19.7	0.04	9.9	0.020

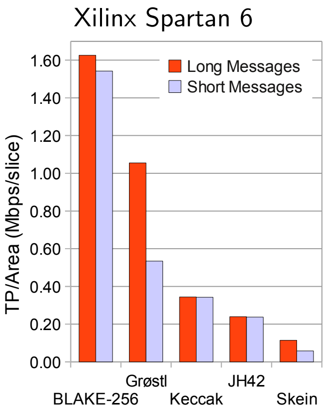
- Short Message : clock cycles associated with initialization, loading, processing, finalization
- No. of blocks of message(N)=1 after padding for short message

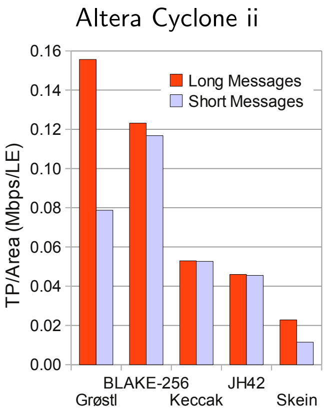
Throughput over Area of 5 Finalists



- No difference in ranking on the two devices.
- Keccak better than JH here, compared to Spartan-3.

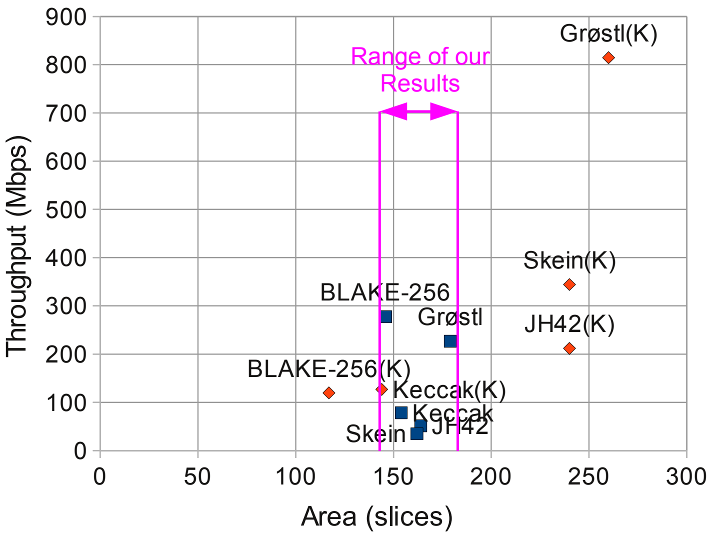
Throughput over Area of 5 Finalists



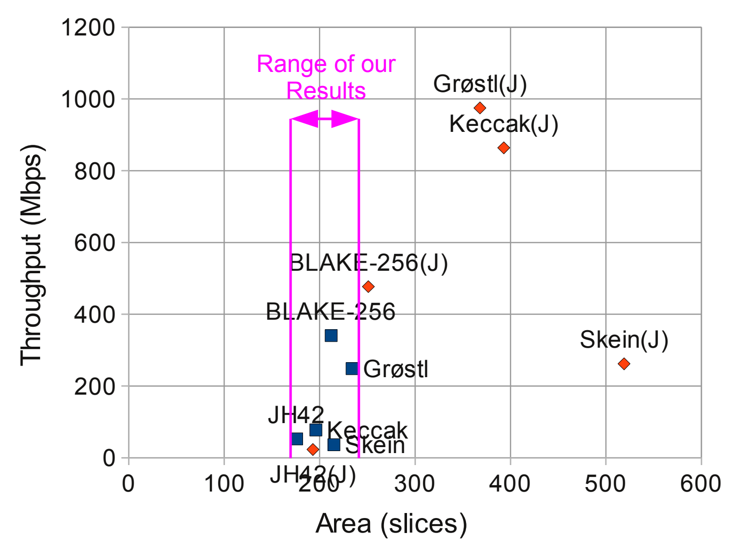


- Grøstl better than BLAKE on Cyclone ii.
- Small changes in ranking depending on device.

Comparison with [Kerckhof] Results (Virtex 6)



Comparison with [Jungk] Results (Virtex 5)



Announcement

- All the above data will shortly be available in the ATHENa database.
  - http://cryptography.gmu.edu/athenadb/
- Source codes will be available on the ATHENa webpage at the end of December.
  - http://cryptography.gmu.edu/athena/
  - Follow the "GMU Source Codes" Link

Thanks

This work has been supported in part by NIST through the Recovery Act Measurement Science and Engineering Research Grant Project under contract no. 60NANB10D004.

Thanks for your attention.