Scalability of SHA-3 Finalists for Lightweight Implementations on FPGAs

Jens-Peter Kaps

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

Cryptographic Architectures Embedded in Reconfigurable Devices, CryptArchi 2011

Outline

Motivation

- NIST Competition to develop SHA-3 $\rightarrow$ 5 finalists.
- Requirement: New function should perform well on a variety of platforms including low resource devices.
- Low-area implementations highlight flexibility of algorithm designs.
- FPGAs are very popular due to low up-front cost and reconfigurability.

Special thanks

- Smriti Gurung (Keccak), Bilal Habib (Skein), Kishore Kumar Surapathi (Grøstl), John Pham (JH), Susheel Vadlamudi (BLAKE), and Panasayya Yalla (low area design).
- This work has been supported in part by NIST through the Recovery Act Measurement Science and Engineering Research Grant Project under contract no. 60NANB10D004.

Previous Work on SHA-3 Candidates

- Several of Throughput/Area optimized implementations on FPGAs were published [Gaj], [Matsuo], [Baldwin].
- Only one specific for low-area implementations of SHA-3 finalists [Kerckhof] (HASH 2011).
  - Not the smallest possible implementation.
  - Area varies from 117 to 304 slices on Virtex 6.
  - Throughput varies from 105 Mbit/s to 960 Mbit/s.
  - No clear design target. Makes comparison difficult.

Goal: Low-area implementations of SHA-3 finalists which:

- All use the same standardized interface.
- All optimized for the same target.
- Explore scalability for lightweight implementations.

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.

Target Details:

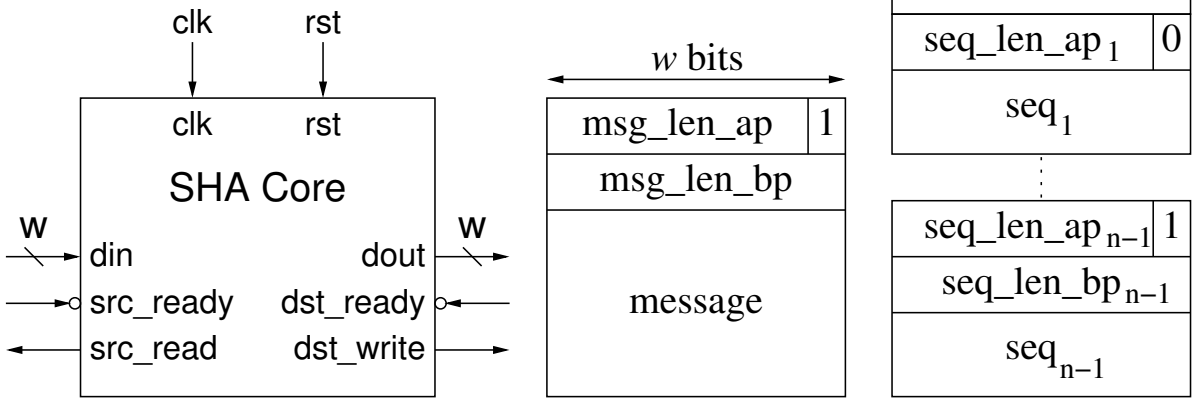
- Xilinx Spartan 3, low cost FPGA family
- Budget: 500 slices, 1 Block RAM (BRAM)

Interface and Protocol

- Based on Interface and I/O Protocol from [Gaj], w=16.
- msg_len_ap, seq_len_ap (after padding) in 32-bit words.
- msg_len_bp, seq_len_bp (before padding) in bits.

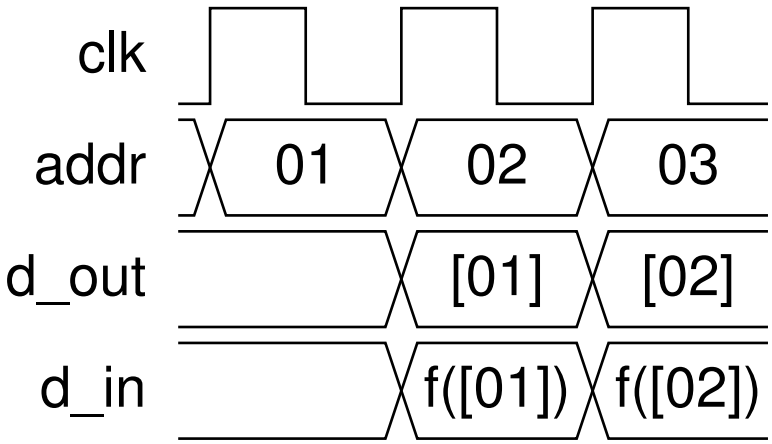
$$msg\_len\_bp = \sum_{i=0}^{n-2} seq\_len\_ap_i \cdot 32 + seq\_len\_bp_{n-1}$$

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



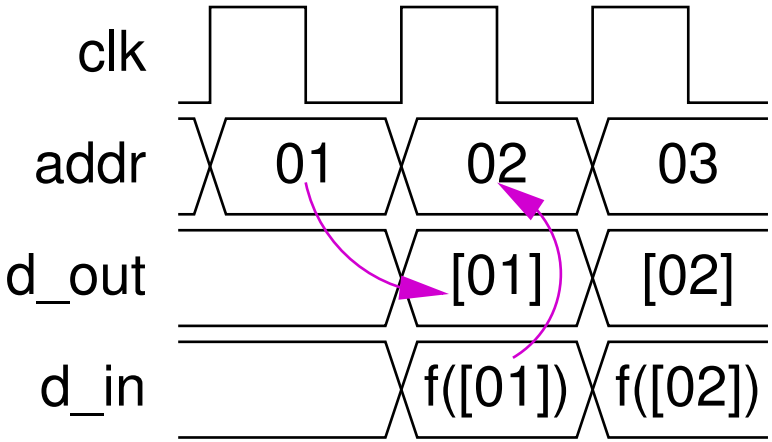
a)SHA Interface b)SHA Protocol

Block RAM



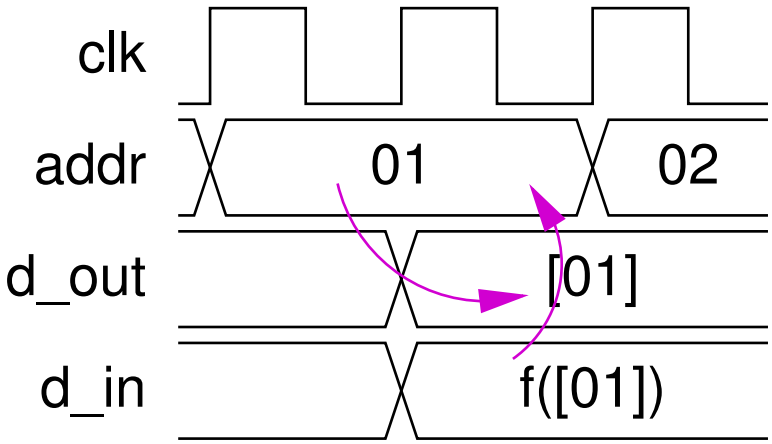
- Compute $A[i] = \mathrm{f}(A[i]), i = 1 \ldots 16$.
- $A[\,]$ is stored in Block RAM.

Block RAM



- Compute $A[i] = \mathrm{f}(A[i]), i = 1 \ldots 16$.
- $A[\,]$ is stored in Block RAM.
- Stores $\mathrm{f}(A[1])$ in $A[2]$.
- Computes $A[i+1] = \mathrm{f}(A[i])$.

Block RAM



- Compute $A[i] = \mathrm{f}(A[i]), i = 1 \ldots 16$.
- $A[\,]$ is stored in Block RAM.
- BRAM needs dedicated read and write clock cycles when same port is used.
- Else the addressing becomes complex.

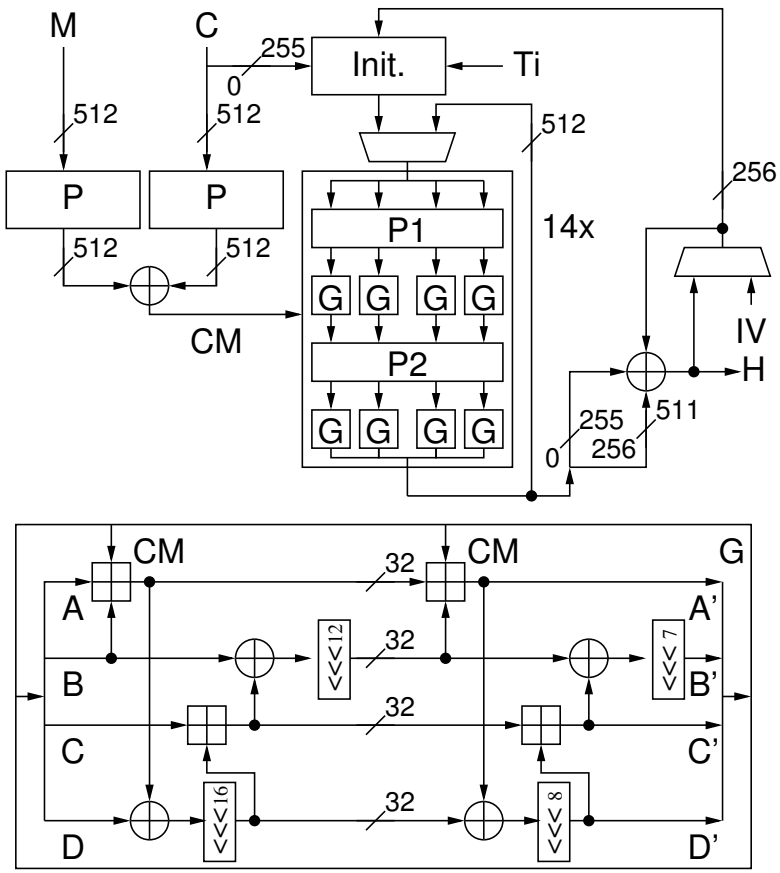
Limits

- Maximum 2 input and 2 output ports, 2 addresses (dual port).
- Maximum single port w/ 64 bits or dual port w/ 32 bits each.

Control vs. Datapath

- Datapath
  - Use BRAM to store state, initialization vectors, constants.
  - Use BRAM in each clock cycle.
  - Avoid temporary storage or use:
    - Free registers, i.e. unused flip-flops after LUTs.
    - Shift Registers (1x16 bit / Distributed RAM (1x16 bit) $\Rightarrow$ 1 LUT $= \frac{1}{2}$ Slice.
- Control Logic
  - Small main state machine, up-to 8 states.
  - Counter for clock cycles in longest state.
  - Stored Program Control within states.
  - BRAM addressing must follow regular sequence, can have offset between rounds.
- Simple datapath might result in complex control unit and vice versa.

BLAKE-256 Algorithm

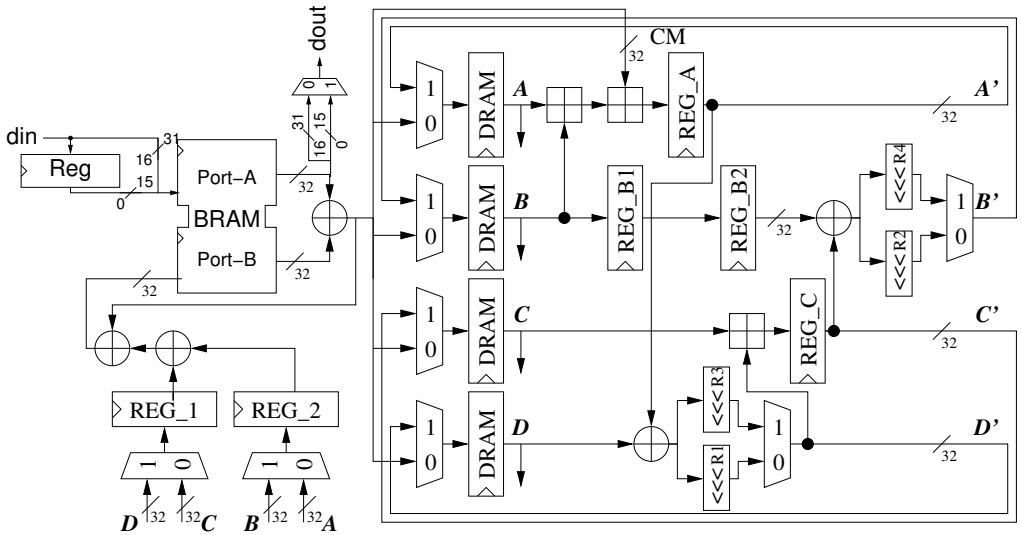


- Blake scales very well.
- Folded up to 4 times vertical and 4 times horizontal.

Memory Requirements: 2,176 bits

- State: 512 bits
- IV: 256 bits
- Intermediate Hash: 256 bits
- Message: 512 bits
- Constants: 512 bits
- Salt: 128 bits all 0

BLAKE-256 Implementation

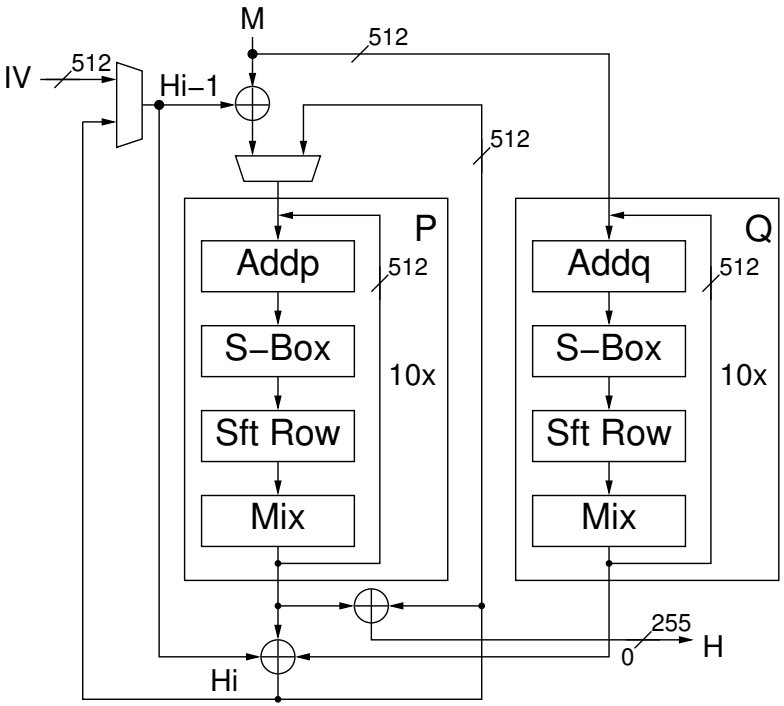


Performance (Clock Cycles)

- Initialization: 2
- Loading: 32
- Block Initialization: 16
- 8xG: 14x21=294
- Block Finalization: 8
- Total per Block: 350

- Implemented pipelined 1/2 G-function computing 2 in parallel.
- 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 clk per block.

Grøstl Algorithm

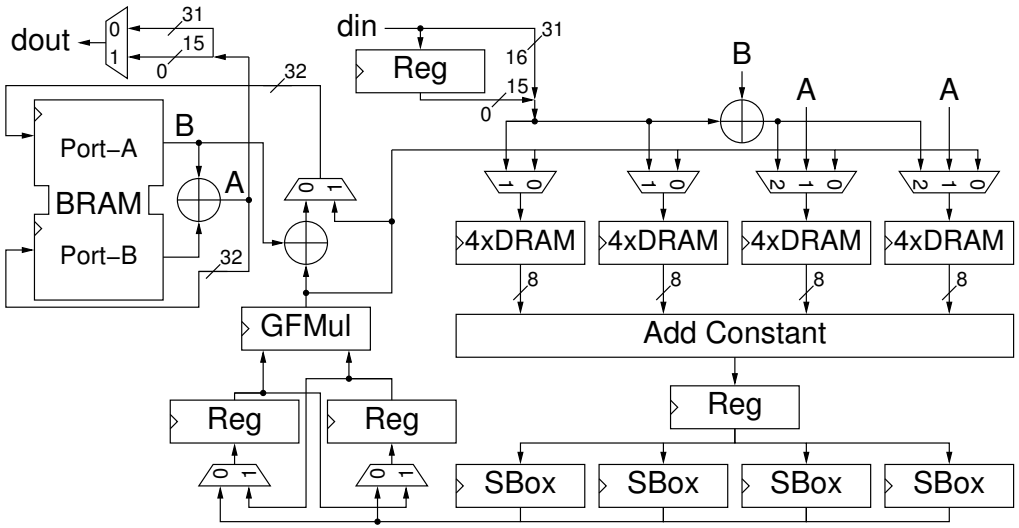


- Grøstl scales well, like AES.
- Folded up to 8 times vertical.
- Small storage requirements.
- Uses many narrow memory accesses in parallel (8 per column).

Memory Requirements: 2,048 bits

- State: 1024 bits
- IV: 512 bits
- Intermediate Hash: 512 bits

Grøstl Implementation

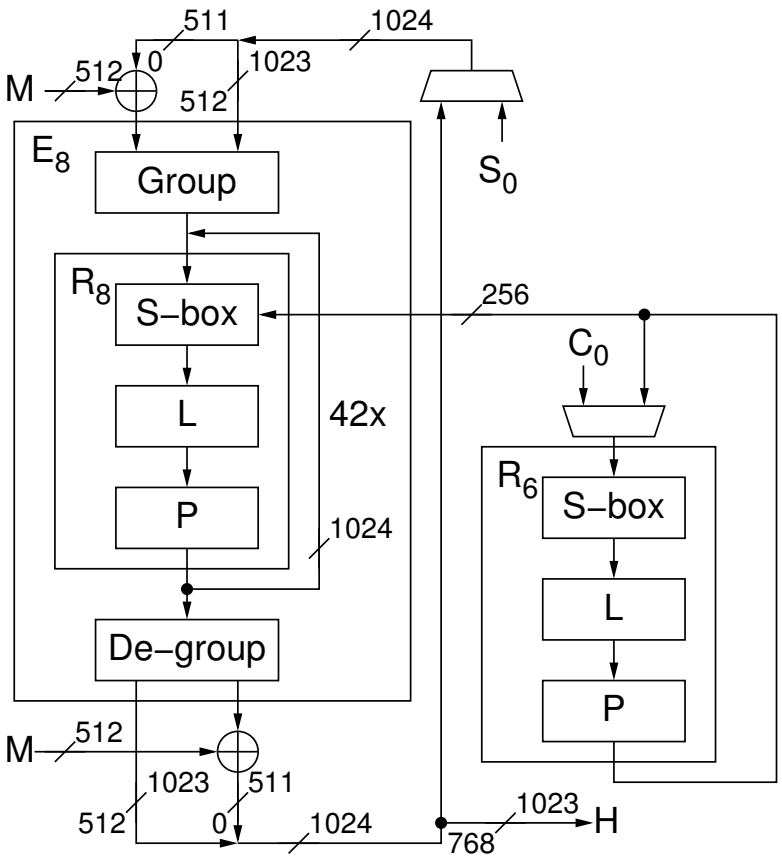


Performance (Clock Cycles)

- Initialization: 2
- Loading: 32
- P: 10x24+3=243
- Q: 10x24=240
- XORs: 32
- Total per Block: 547

- 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

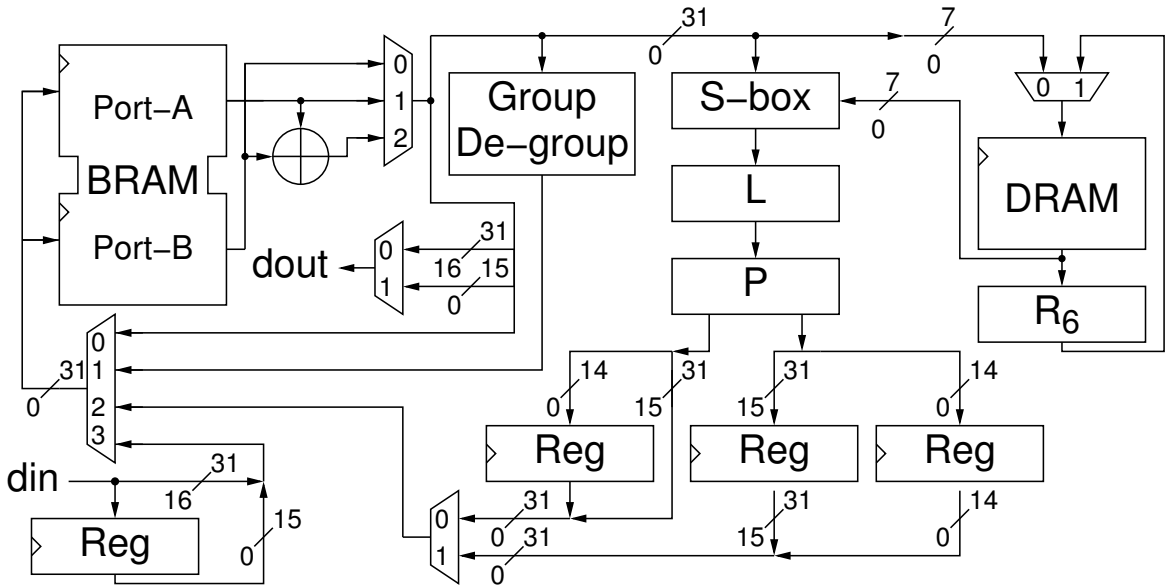


- Permutation P, grouping, and de-grouping makes scaling difficult.
- Folding increases size.

Memory Requirements: 3,072 bits

- State: 1024 bits
- Precomputed S_0: 1024 bits
- Message: 512 bits
- Constants State: 256 bits
- Constants C_0: 256 bits

JH Implementation

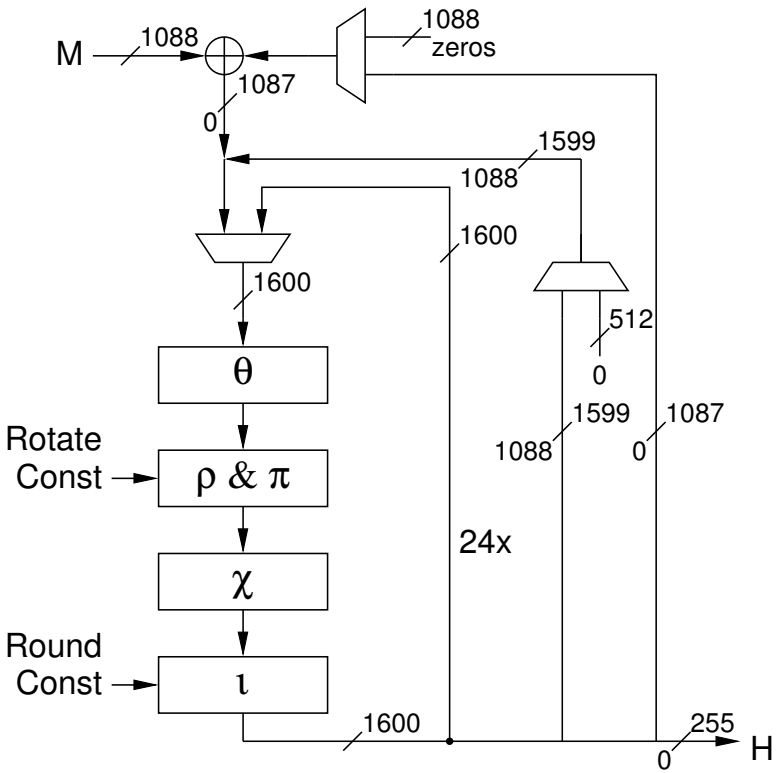


Performance (Clock Cycles)

- Initialization: 35
- Loading: 32
- Group/De-group: 320
- R_8: 34x42=1,428
- Message XOR: 65
- Total per Block: 1,845

- 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].

Keccak Algorithm

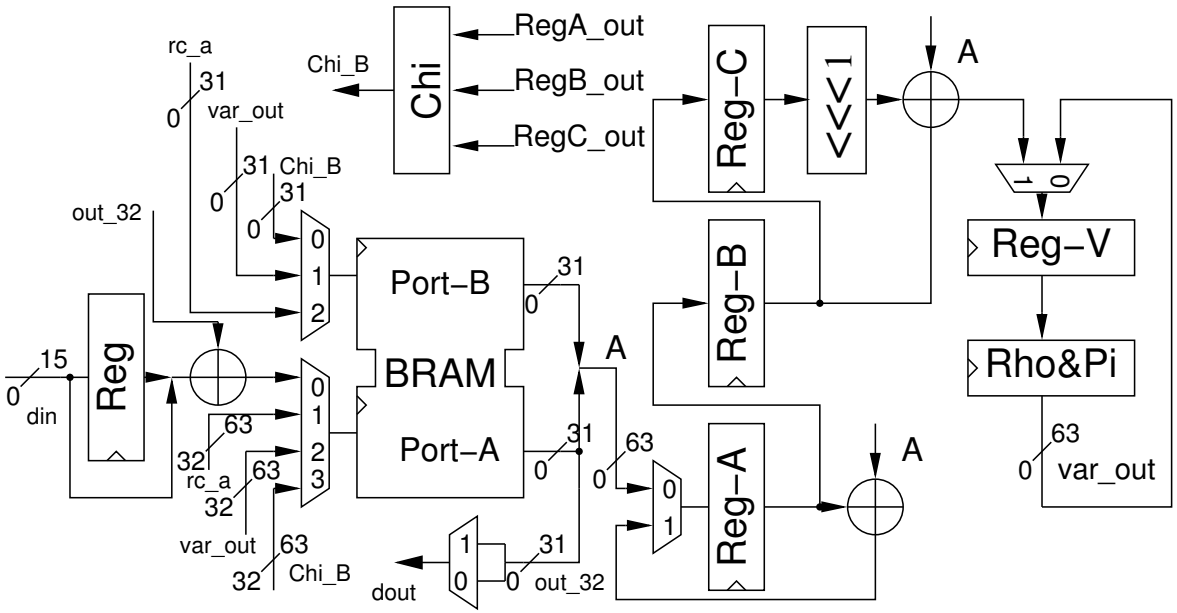


- Only simple operations.
- ρ, π operate on columns, χ on rows.
- Algorithm cannot be folded.

Memory Requirements: 3,286 bits

- State: 1,600 bits
- Round Constants: 1,536 bits
- Constants: 150 bits

Keccak Implementation

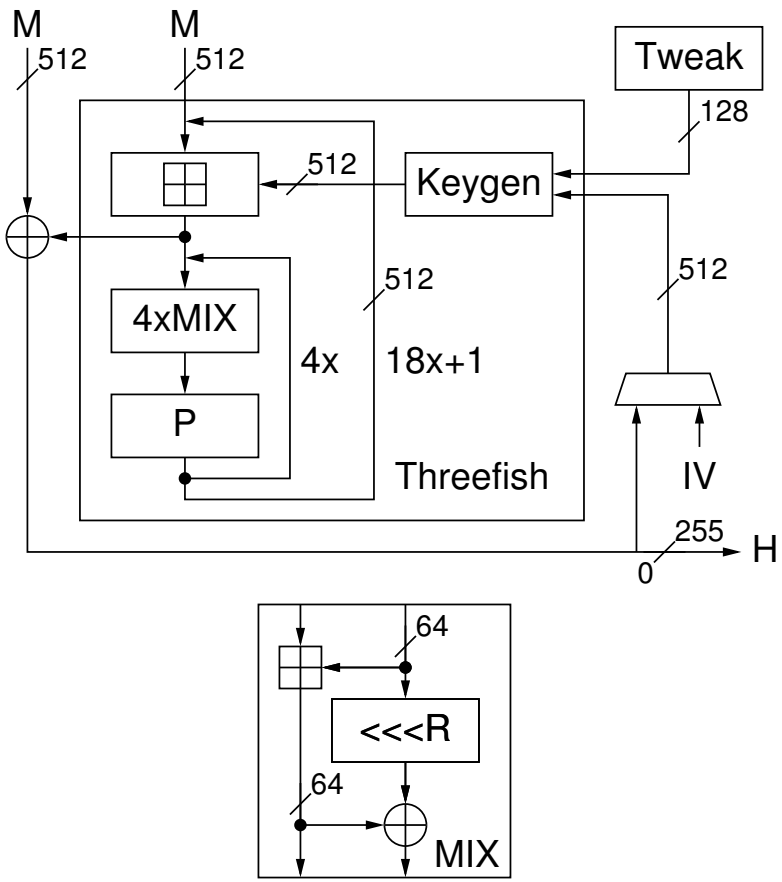


Performance (Clock Cycles)

- Initialization: 2
- Loading: 68
- θ, ρ, π: 91x24=2,184
- χ, ι: 63x24=1,512
- Total per Block: 3,764

- Fixed rotations turn into variable rotator for small datapaths.
- ρ & π contains the rotator.
- **Scalability:** 64-bit datapath only viable without BlockRAM.
- Adding 2 more 64-bit registers saves approx 700 clock cycles.

Skein Algorithm

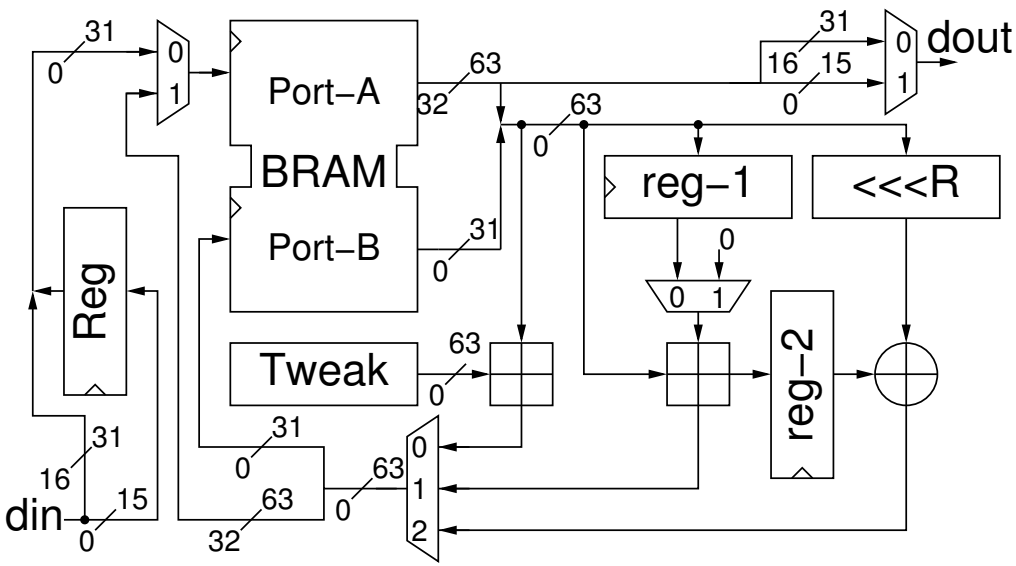


- 64-bit adders lead to long delay.
- Algorithm cannot be folded.

Memory Requirements: 2,112 bits

- State: 512 bits
- Processed IV: 512 bits
- Message: 512 bits
- Rotation Tweak Constants: 64 bit
- Hash Chaining Value: 512 bits

Skein Implementation



Performance (Clock Cycles)

- Initialization: 5
- Loading: 32
- MIX: 18x4x20=1440
- Keygen: 19x62=1178
- Total per Block: 2650

- 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:** 32-bit adder reduce critical path and have no clock cycle penalty.

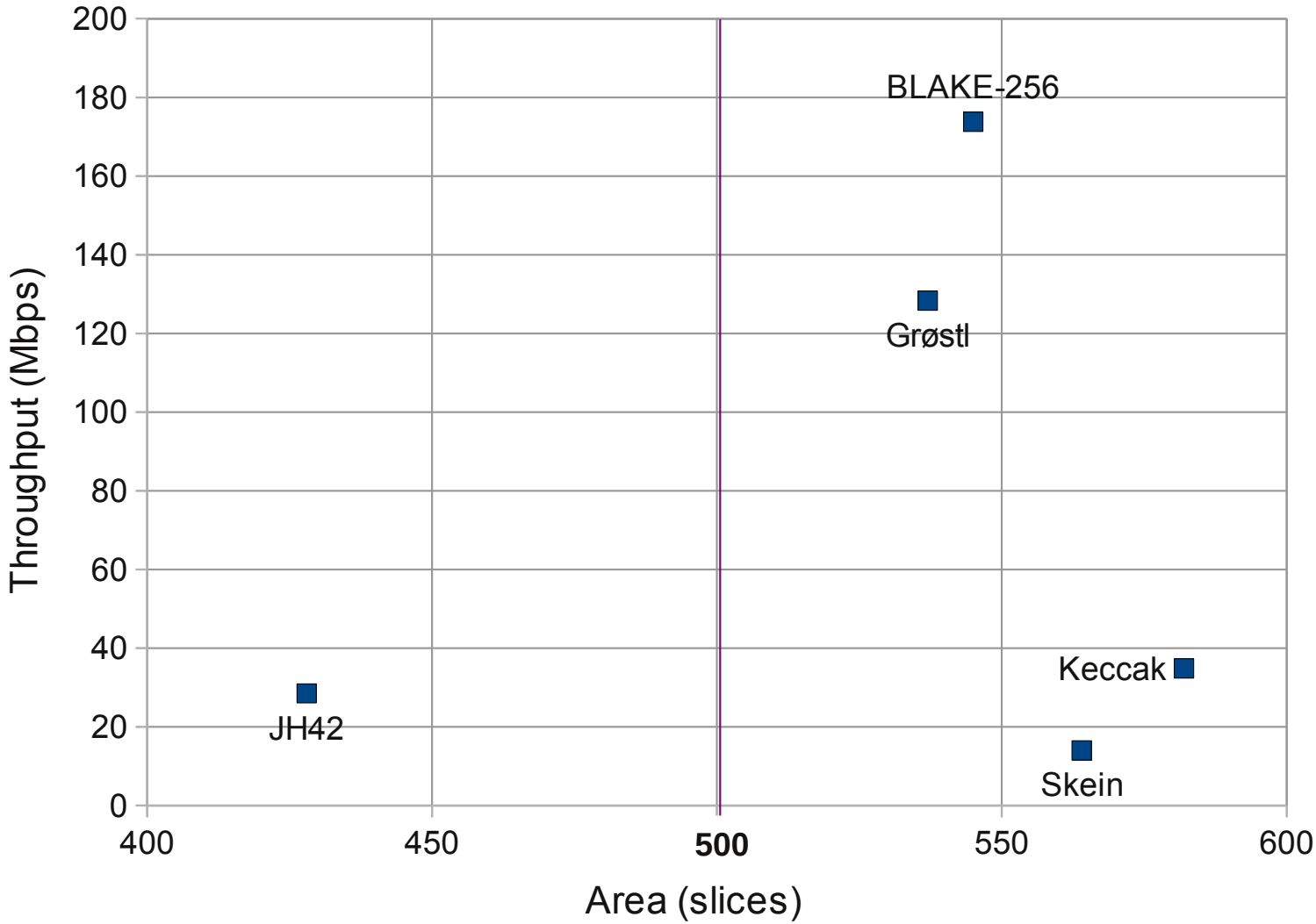
Implementation Summary

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	$2 + (\ 32 + \ 542) \cdot N + \ 559$	$512/(\ 574 \cdot T)$
JH42	512	$35 + (\ 32 + 1813) \cdot N + \ 17$	$512/(1845 \cdot T)$
Keccak	1088	$2 + (\ 68 + 3696) \cdot N + \ 17$	$1088/(3764 \cdot T)$
Skein	512	$5 + (\ 32 + 2428) \cdot N + 2444$	$512/(2460 \cdot T)$

Results on Xilinx Spartan-3

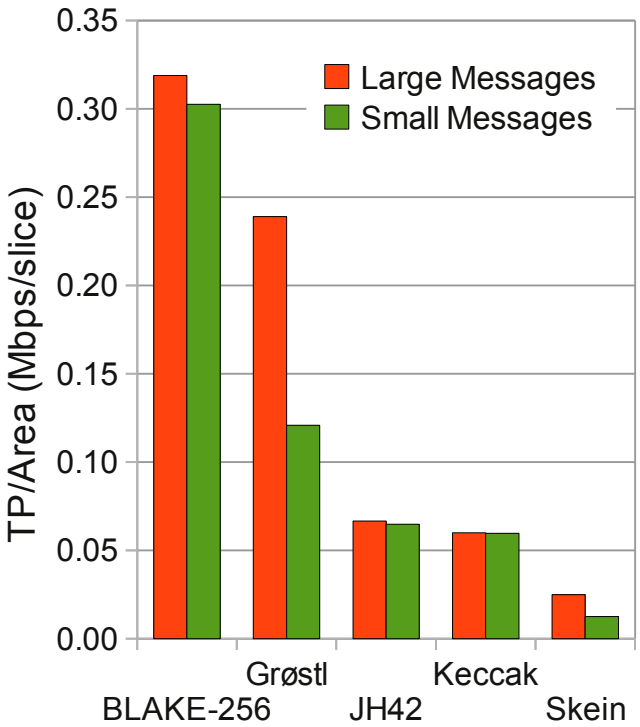
Algorithm	Area (slices)	Block RAMs	Maximum Delay (ns) T	Long Message		Short Message	
				Throughput (Mbps)	TP/Area (Mbps/slice)	Throughput (Mbps)	TP/Area (Mbps/slice)
BLAKE-256	545	1	8.42	173.8	0.32	164.8	0.302
Grøstl	537	1	6.95	128.3	0.24	64.9	0.121
JH42	428	1	9.74	28.5	0.07	27.7	0.065
Keccak	582	1	8.30	34.8	0.06	34.7	0.060
Skein	564	1	14.85	14.0	0.02	7.0	0.012

Throughput over Area on Spartan 3

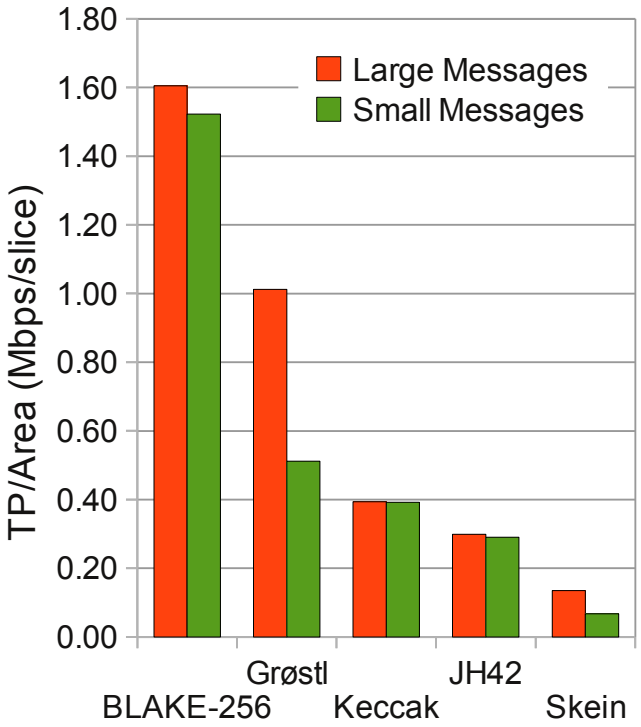


Throughput over Area

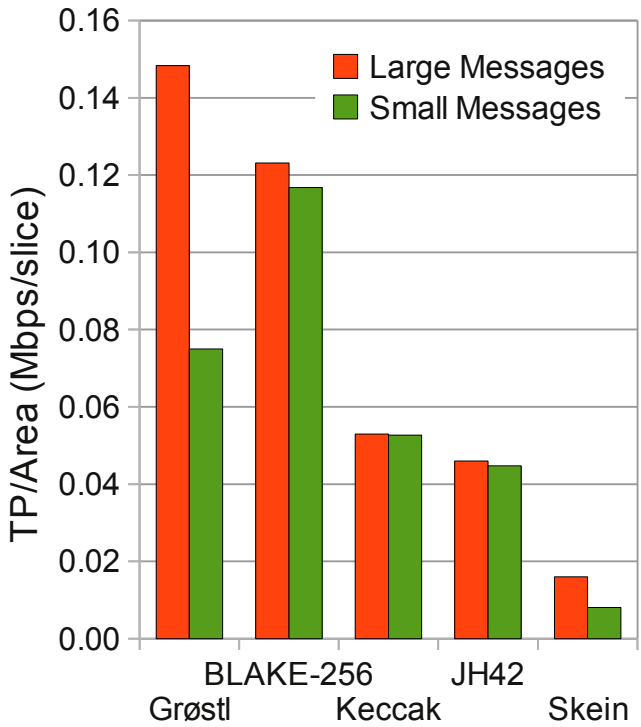
Xilinx Spartan-3



Xilinx Virtex 5

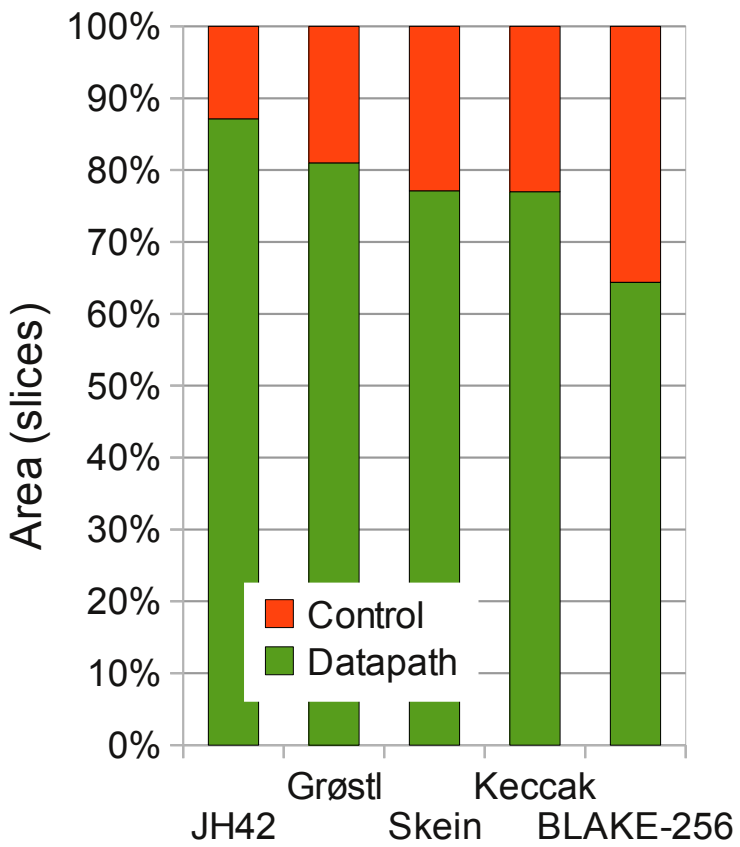


Altera Cyclone ii



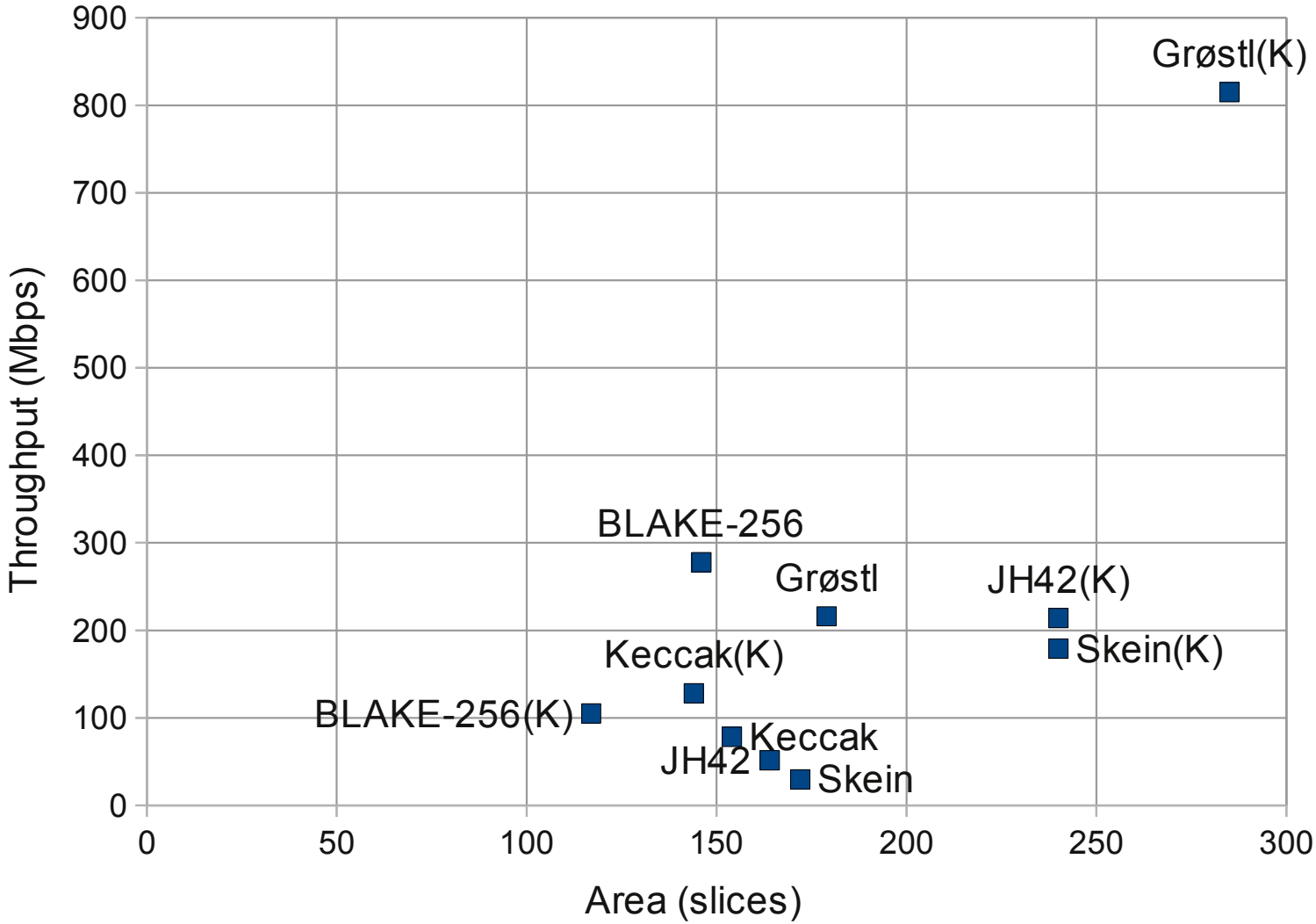
- Algorithms with finalization rounds perform worse for small messages.

Datapath vs. Control



- Number of clock cycles per block is not an indication for control logic size (Blake: 350, JH: 1845).
- Blake has a permutation schedule with 210 entries.
- JH allows for a very regular schedule of control signals.
- Size of control logic depends on:
  - Number of control signals
  - Number of stages/functions
  - Number of clock cycles per function
  - Schedule of constants
  - Regularity of addresses for storage

Comparison with [Kerckhof] Results (Virtex 6)



Thanks for your attention.