Uniform Evaluation of Hardware Implementations of the Round-Two SHA-3 Candidates

Stefan Tillich[1,2], Martin Feldhofer[1], Mario Kirschbaum[1], Thomas Plos[1], Jörn-Marc Schmidt[1], and Alexander Szekely[1]

1) IAIK – Graz University of Technology
www.iaik.tugraz.at

2) University of Bristol
tillich@cs.bris.ac.uk

University of BRISTOL

Implementation Factors

- Design target
  - High throughput
  - Low area
  - Low power
- Functionality
- Interface
- Synthesis tools
- Target technology; standard-cell library
- Optimization heuristics

SHA-3 HW Benchmark @ IAIK

- http://www.iaik.tugraz.at/content/research/vlsi/sha3hw/
- All 14 candidates implemented by VLSI team
- Goal: Capture “core” properties of candidate for HW implementation
  - Not the properties of a specific interface

Uniform Implementation

- 256-bit message digests
- Plain hashing only (no salting, etc.)
- Aimed for high peak throughput
- “Broad” generic interface
- Padding external
- Same design tools and technology/standard cells
- Synthesis under worst-case conditions
- Same optimization heuristic (multiple synthesis runs with automatically adjusted delay target)

Performance Metrics

- Peak throughput (in Gbit/s)
  - Long messages (neglect initialization/finalization overhead)
  - Maximum clock frequency @ typical conditions
- Post-synthesis area (in GEs)
- Post-P&R area (in GEs)
- Layout area (in sqmils)
- Implementation towards those metrics
  - Multiple area-performance trade-offs/design options tested

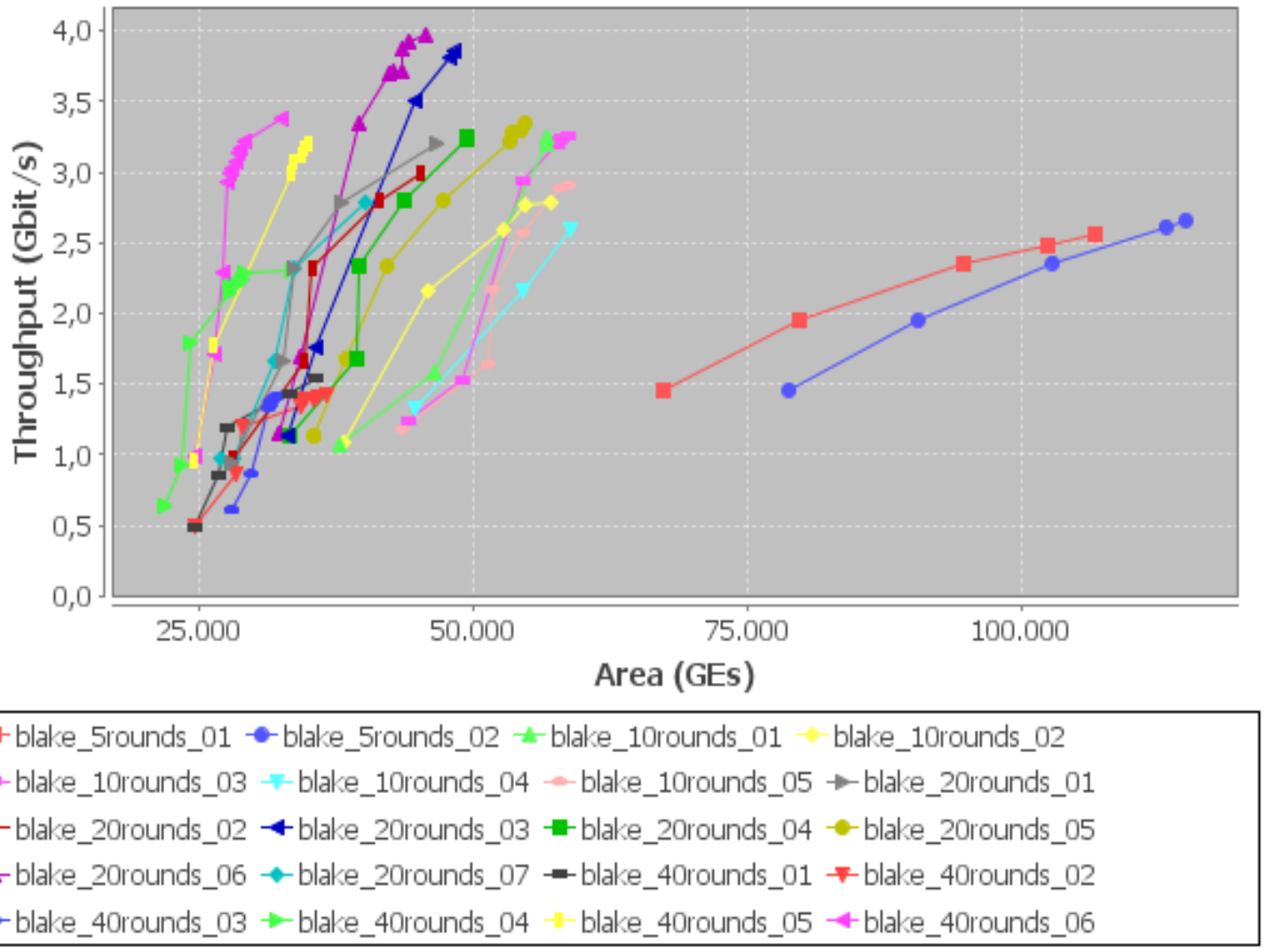
Implementation Strategy

- Six implementers (medium to experienced)
- Predominantly VHDL, two designs in Verilog
- Explored different performance/area-trade-offs/design options
- Multiple synthesis runs (adjusting delay target)
- Picked fastest design options with “reasonable” area requirements
- Perform P&R for that option

Workload

- Manual pre-selection of worthwhile design options
  - Using “manual” synthesis runs
- Series of automatic synthesis runs for each selected variant
  - Running time limited to 2 hours each
  - Hundreds of individual runs in total
- Place & Route (P&R) of “best” variants
  - “Best”: Highest peak throughput; any further *x*% increase of throughput (if any) would cost at least *10x*% extra area
  - No time limit

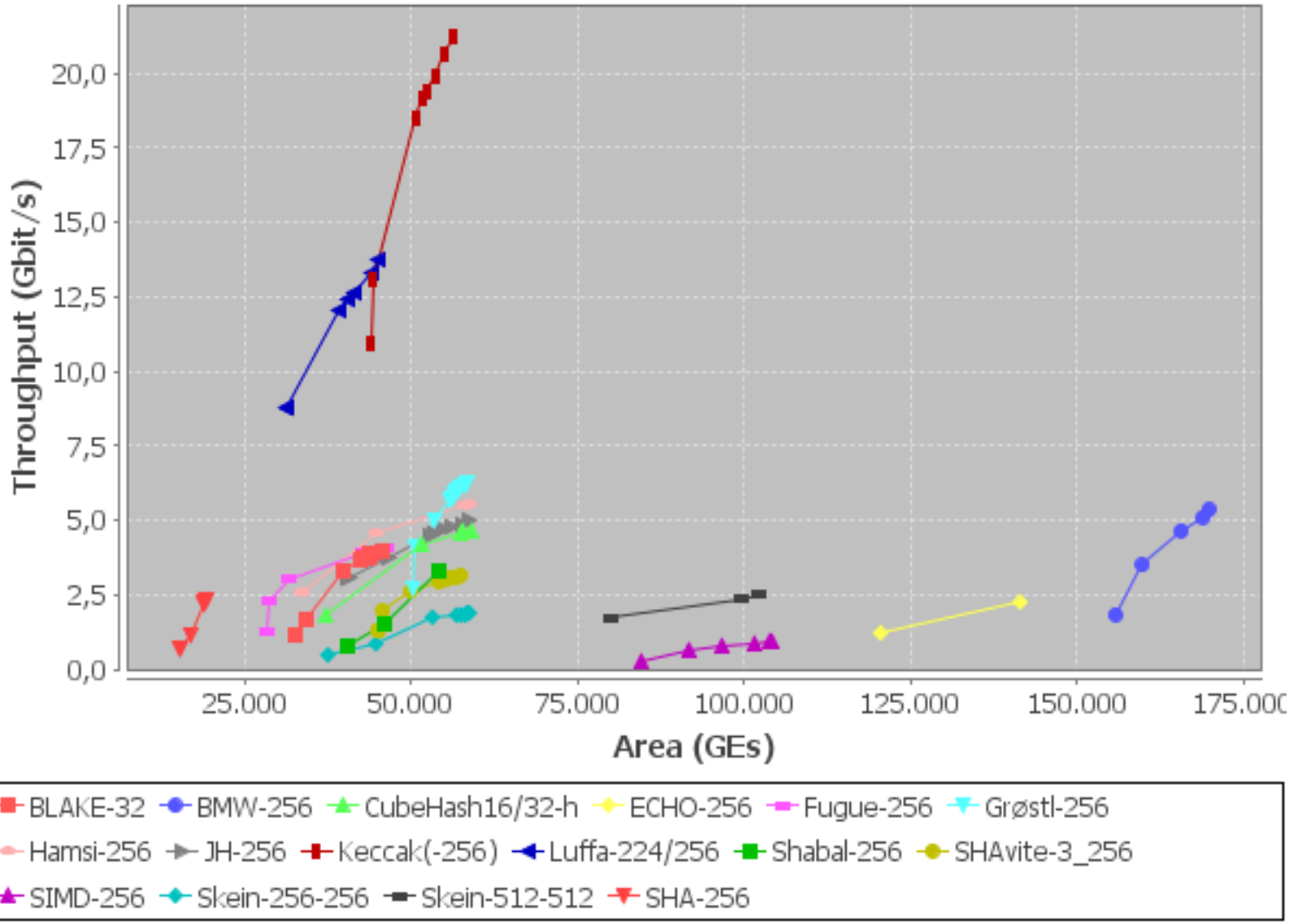
Synthesis Runs: BLAKE-32



Design Optimization

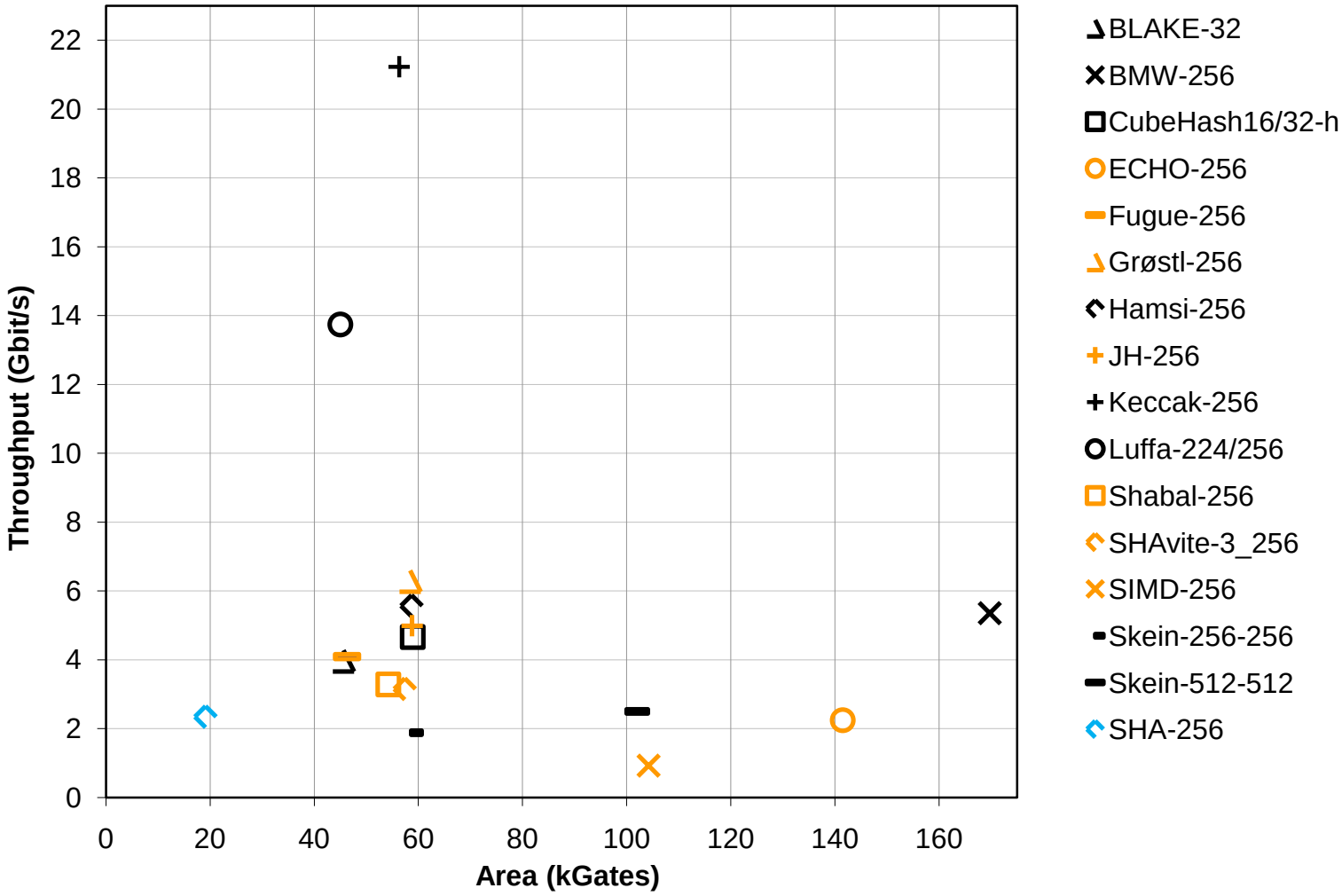
Candidate	# design variants	# synthesis series
Blake-32	112	20
BMW-256	3	3
CubeHash-16/32-h	5	1
ECHO-256	2	2
Fugue-256	3	3
Grøstl-256	34	2
Hamsi-256	4	1
JH-256	5	2
Keccak(-256)	1	1
Luffa-224/256	6	4
Shabal-256	1	1
SHAvite-3_{256}	24	3
SIMD-256	1	1
Skein-256-256	2	1
Skein-512-512	2	1
SHA-256	2	2

Post-Synthesis Results



Interactive version: http://www.iaik.tugraz.at/content/research/vlsi/sha3hw/

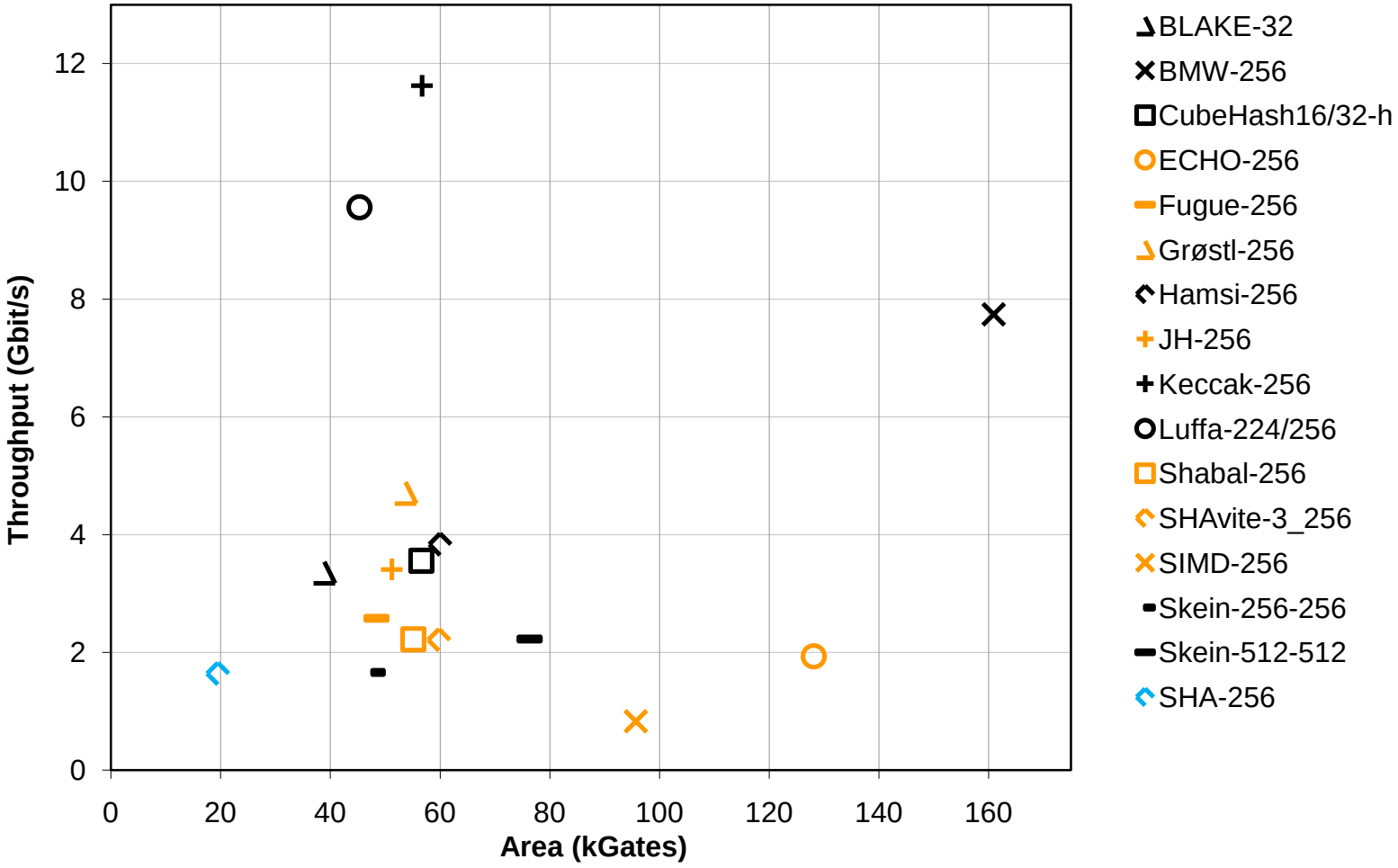
Post-Synthesis Results: “Best” Implementations



Synthesis -> Place & Route

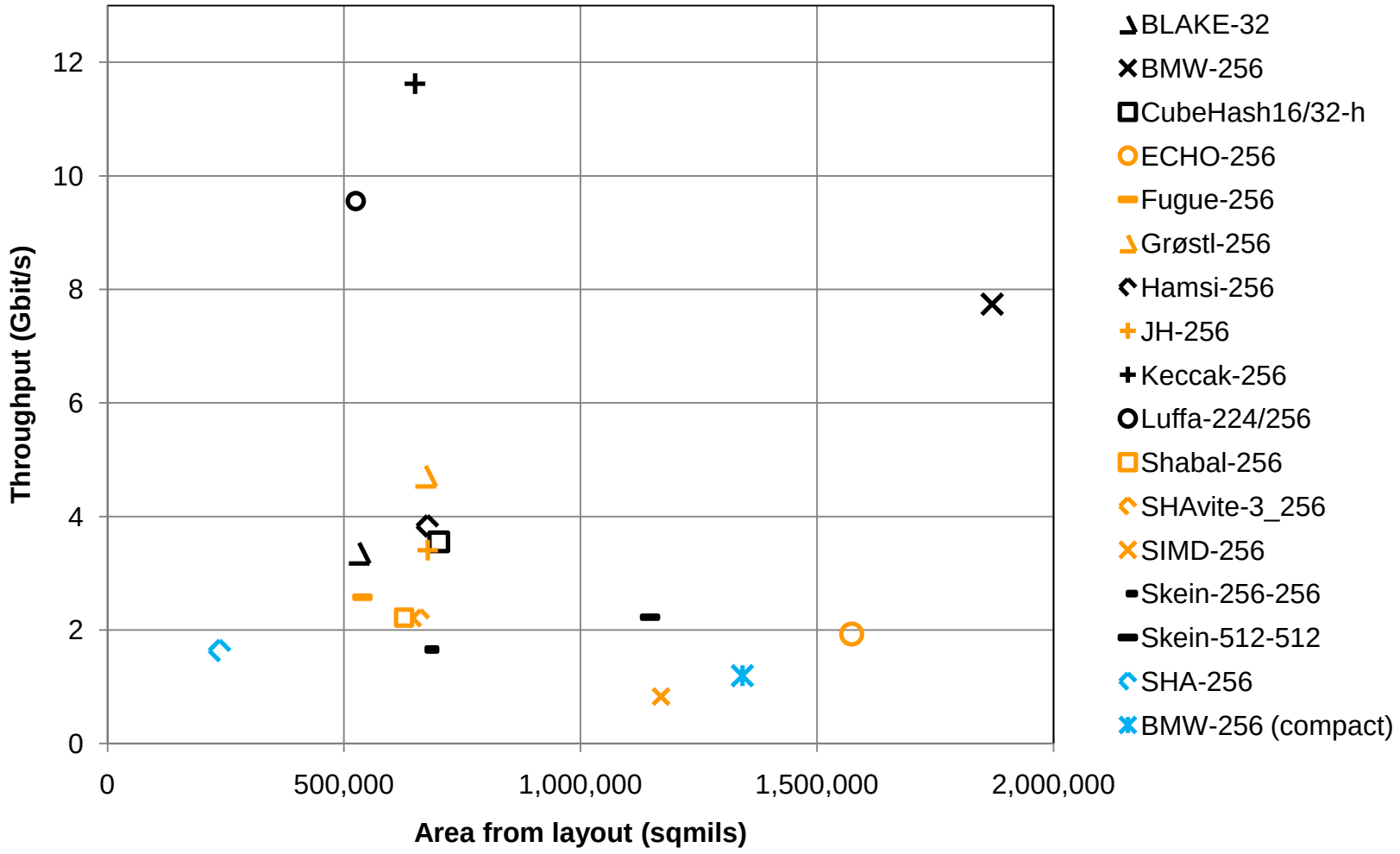
- Post-synthesis result
  - Netlist contains extra buffers
  - Critical path delay estimated by synthesis tool
- Post-P&R result
  - Most of the buffers removed -> slightly less gates
  - Critical path increases -> reduction in throughput (gap narrows)

Post-P&R Results (Gate Count)



Note: Gate count does not include overhead due to clock tree, power nets, and routing

Post-P&R Results (Layout)



Note: Area from layouts includes overheads due to clock tree, power nets, and routing

Some Observations

- Post-P&R (gate count, layout) in good accordance
- No impls as small as SHA-256
- Almost all impls outperform SHA-256
- Keccak and Luffa impls somewhat ahead
- Many impls in a similar range: Grøstl, Hamsi, CubeHash, JH, BLAKE, Fugue, Shabal, SHAvite-3, Skein
- Some impls considerably larger: SIMD, ECHO
- BMW impl fast but large

What it says

- Lower bounds for each candidate's performance
  - -> Improvements always possible
- Approximate upper bounds for area
  - -> Smaller with less performance possible
- Rough relative comparison of candidates for given design target (peak throughput)
  - -> Concrete design choices might change picture, e.g. supported functionality (padding, multiple MD sizes, multi-message hashing), interface

What it does not say

- Candidate *x* is generally better than candidate *y*
- Candidate *x* will require exactly *y* gates to reach *z* Mbit/s on technology *t*

Thanks for your attention!