SHA-3 optimization and benchmarking

Christian Wenzel-Benner

Outline

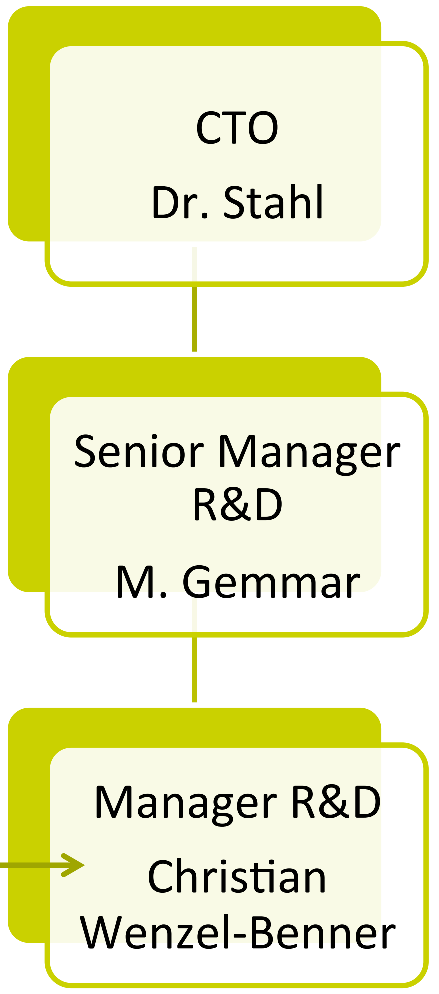
Sections of this talk

Who‘s talking?

Christian Wenzel-Benner

- I‘m an engineer
  - Dipl-Ing.(BA) Computer Engineering
  - M.Sc. Distributed Computing Systems Engineering
- I‘m interested in hashes
  - Needed them for my master‘s dissertation
  - Followed the SHA-3 competition since 2009 as a hobby together with Jens Gräf
- I have a daytime job
  - Manager R&D

ITK Engineering



What to expect?

Something for everyone (pick what you like)

- Will cover the fundamentals of benchmarking and optimization in the SHA-3 selection process
- Necessarily rather broad than deep
- 4 different target families – giving a glimpse into a lot of work by other people and groups (look it up – it's good)
- PC and embedded systems benchmarking will be covered in more detail
- If the size of the audience permits feel free to ask really short questions directly, please keep longer ones for Q&A or later

General considerations about (crypto) benchmarking

What exactly is a benchmark?

- A benchmark is the act of running a computer program to assess performance of
  - Different hardware the program runs on
  - Different implementations of the software on the same hardware
  - The term benchmark is also used for software that is specifically designed for performance assessment
- A test suite is a piece of software designed to assess correctness
- Distinction not always useful, i.e. Prime95:
  - Is designed to search for Mersenne prime numbers -> application
  - Can be used to compare CPU performance -> benchmark
  - Can be used to stress the CPU and RAM (a.k.a. torture test) -> test suite

General considerations about (crypto) benchmarking

Challenges and issues of benchmarking (very incomplete list)

- A synthetic benchmark covers only a specific use case
  - Can be very useful to assess and subsequently optimize a particular component of the system being benchmarked
  - Usually not very close to an actual user's experience
- An executable only benchmark is well specified but a black box
- An open source benchmark performs differently w.r.t. compilers, compiler settings, compiler versions but is transparent
- A pure benchmark assesses performance only
  - If the component being benchmarked performs fast but incorrectly this will result in a 'good' benchmarking result
  - Real word example: some graphics card drivers decrease rendering quality when a known benchmark software is detected in order to score higher frame rates

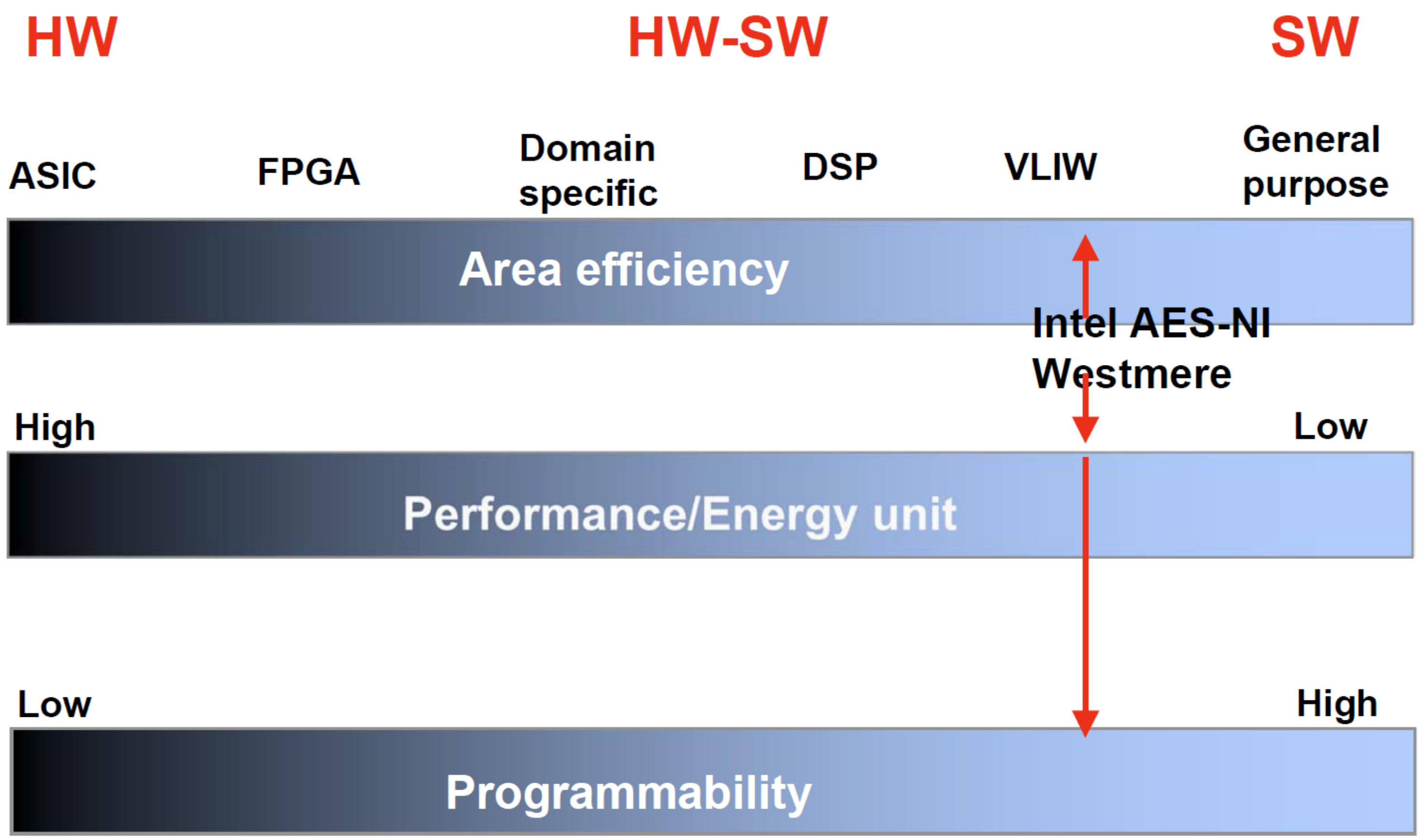
General considerations about (crypto) benchmarking

A "good" benchmark…

- Closely resembles real world performance
  - The closer it gets to actual user experience, the better
- Generates results that can be easily reproduced
  - Both running and understanding the benchmark should be possible for everyone interested in the matter
- Makes it hard for developers to make their product perform well in the benchmark and not in reality
  - Ideally, the benchmark forces the developers to make their products perform better for the user in order to score higher at the benchmark
  - Besides testing for speed and/or memory consumption it should also tests correct functionality

Introduction to benchmarking & implementation of SHA-3

SHA-3 is to be used everywhere from ID tags to PCs



Source: Ingrid Verbauwhede, KU Leuven: Hardware benchmarking for HASH³

Introducing the PC – microprocessor based

- The PC needs no introduction, the PC is everywhere
- What it does need to function however is a lot of components
  - CPU (microprocessor)
  - Mainboard
  - RAM
  - Power supply (several voltage levels)
  - And either
    - Network interface (for a server)
  - Or
    - Screen, keyboard, mouse (for a local workstation)
  - Special I/O cards if you want to measure temperatures, etc.

Introduction to benchmarking & implementation of SHA-3

On the PC – why?

- PCs are general purpose computing devices with graphics and sound and a well established user interface
- They are easily programmable in **several different languages**
- That makes applications available for a lot of usage scenarios
- This makes PC hardware cheap through mass production
- That makes them available almost everywhere on the planet
- This makes the PC a very important platform for SHA-3, despite its low energy efficiency
- A new PC or Laptop can be bought for less than 300€, a used one nearly for free if it is old enough
  - If you want to implement SHA-3 on a PC, it‘s not hard

Introduction to benchmarking & implementation of SHA-3

On the PC benchmarking is easy and straightforward... NOT

- Performance numbers for SHA-2 on the NIST reference PC presented at the 1st SHA-3 Conference in Leuven 2009 (normalized, cycles per byte)
  - SHA-256: 20.1, 40.65/39.1, 21/16,
  - SHA-512: 13.1, 63/13
  - SHA-2, length not specified: 22/20, 41/13
- Can you find the **Truth**™?
- What questions would you ask to the presenters?
- What happened at the conference?
  - DJB

Introduction to benchmarking & implementation of SHA-3

On the PC

- D. Bernstein, T. Lange et. al. (VAMPIRE Lab):
- **S**ystem for **U**nified **P**erformance **E**valuation **R**elated to **C**ryptographic **O**perations and **P**rimitives (SUPERCOP)
  - http://bench.cr.yp.to/supercop.html
  - Benchmarking toolkit, implementations and HUGE results database all in one place
  - Runs on linux, even on some smartphones and tablets
  - Originally intended for PCs and workstations
  - Completely free (as in speech) and open source software
- SUPERCOP benchmarks a lot of crypto algorithm families
  - The part dedicated to hashes is called **E**CRYPT **B**enchmarking of **A**ll **S**ubmitted **H**ashes (eBASH)

Introduction to benchmarking & implementation of SHA-3

On the PC – SUPERCOP handling of: sources, compilers, options

- SUPERCOP enforces maximum transparency
  - Crypto algorithms have to be submitted as source code
  - Anyone is encouraged to download the SUPERCOP package (benchmark scripts + algorithm source codes) and run it on their own PC
- Good: SUPERCOP benchmark numbers are widely accepted
- Bad: Compiler, compiler options, etc. have large influence
- Solution:
  - build executables with all compilers on system
  - use a huge list of compiler options
  - use all available implementations of a specific algorithm
  - find fastest executable in all the permutations of the above
  - measure that executable at many different message lengths

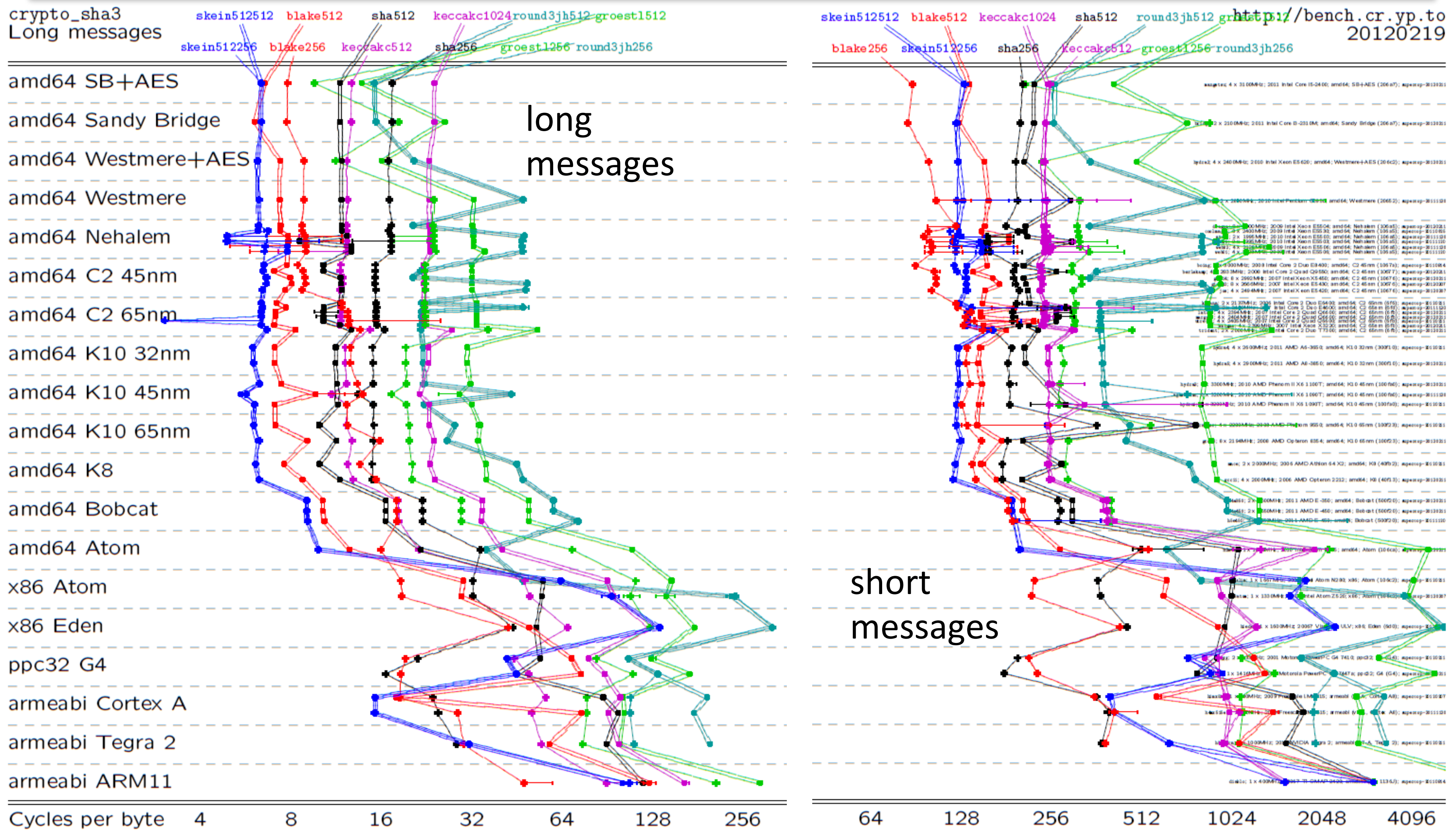
Introduction to benchmarking & implementation of SHA-3

On the PC – SUPERCOP handling of: CPUs, ABIs, clock, multitasking

- Trade names of CPUs change independently of the actual chip
  - SUPERCOP records CPUID and stepping in benchmark results
- Same CPU can run different instruction sets / ABIs
  - SUPERCOP records instruction set and ABI used
  - Also records exact version of compiler used to build executable
- Same CPU can run at different clock rates
  - SUPERCOP records clock rate
  - Measured timings are normalized for clock rate
- PC usually does stuff in the background
  - CPU may be taken away from crypto benchmark by OS
  - SUPERCOP records thousands on executions, reports median + quartiles

Introduction to benchmarking & implementation of SHA-3

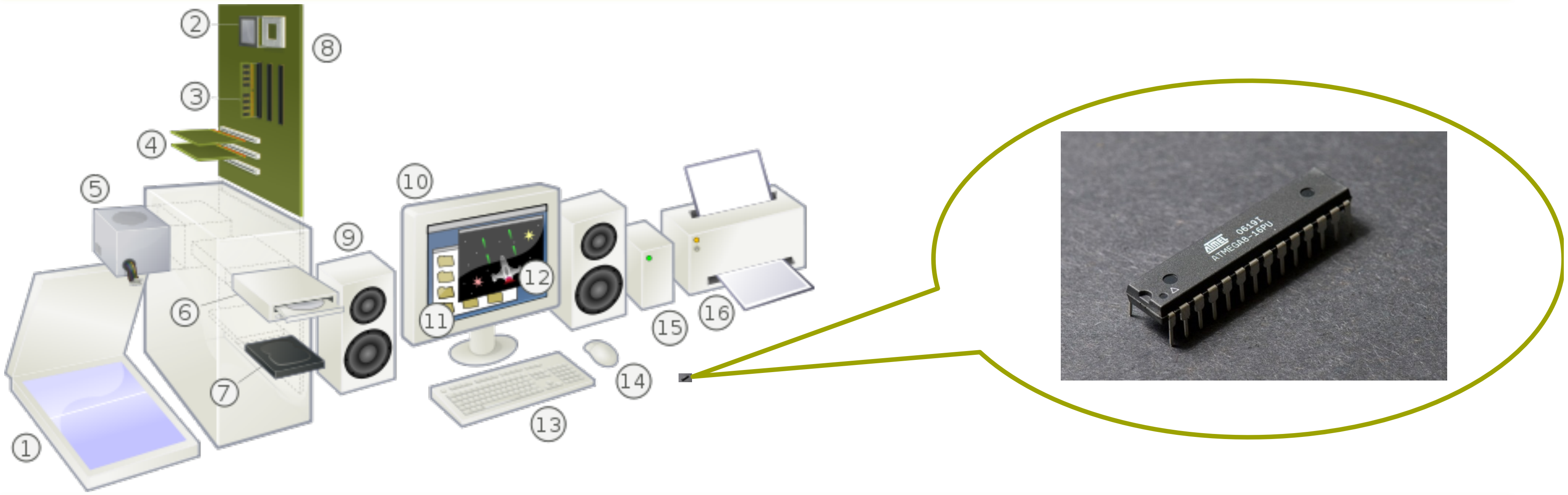
On the PC



Introduction to benchmarking & implementation of SHA-3

Introducing embedded platforms – microcontroller based

- As opposed to a PC, a microcontroller needs very few extra components and is very small and cheap
  - Some work when provided with a single supply voltage and nothing else
  - Interaction with electronics like temperature sensors usually do not need additional interface components



Source: Wikimedia commons, CC-BY-SA http://en.wikipedia.org/wiki/File:Personal_computer,_exploded_6.svg , commons.wikimedia.org/wiki/File:ATmega8_01_Pengo.jpg

Introduction to benchmarking & implementation of SHA-3

On embedded platforms – why?

- Embedded platforms are small, self-contained computers
- They are an integral part of many modern machines
  - Smart phones, tablets
  - Dishwashers, fridges, ovens
  - Internet / DSL / cable routers
  - Automobiles, motorbikes
  - Planes, rockets, satellites, ground and space based telescopes, …
- Obviously we need SHA-3 on quite a few of those
- Programming usually requires C or assembly language
- The prices for development kits start at around 15€
  - If you want to implement SHA-3 on an embedded platform, it's not expensive but you need some not-so-common skills like low level, low memory footprint programming

Introduction to benchmarking & implementation of SHA-3

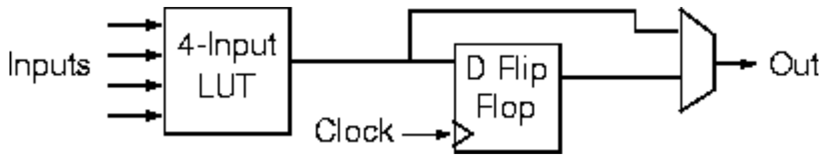
On embedded platforms

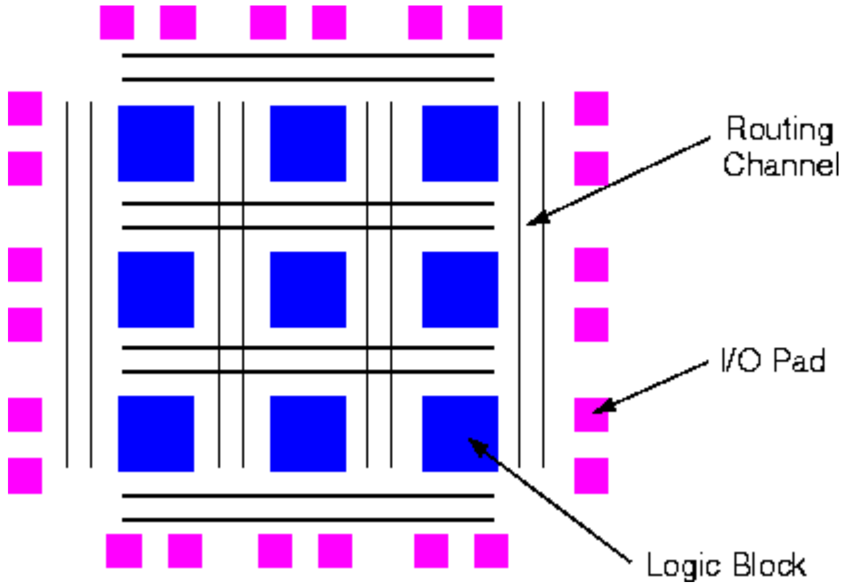
- C. Wenzel-Benner, J. Gräf et. al. (XBX Team):
- e**X**ternal **B**enchmarking e**X**tension (XBX) extends SUPERCOP-eBASH to use embedded platforms as benchmarking targets
    - Actually, the XBX code differs quite a bit from SUPERCOP
    - But it reads the same source format
    - Adheres to the same benchmarking philosophy
    - Outputs the same results format (plus extensions)
    - And delivers memory consumption results in addition to timing
- More details on XBX and implementations in the next section

Introduction to benchmarking & implementation of SHA-3

Introducing FPGAs – Field Programmable Gate Arrays

- FPGAs contain a larger number of logic gates
- They are bundled in small packages called “logic blocks”, “Slices” or “ALUTs”
- The user can connect those packages (almost) at will
- Additionally there are some “special” packages like RAM blocks, DSPs, I/O interfaces





Source: Vaughn Betz , University of Toronto, http://www.eecg.toronto.edu/~vaughn/challenge/fpga_arch.html

Introduction to benchmarking & implementation of SHA-3

On FPGAs – why?

- **F**ield **P**rogrammable **G**ate **A**rrays (FPGAs) are chips that provide the implementer with a lot of logic gates that can be programmed and re-programmed at will ("field programmable")
- They are mainly used for two reasons
  - To implement things that run best in hardware yet either are not finally specified or won't sell enough units to justify building a dedicated chip
  - To start early development on a dedicated chips functionality and estimate achievable performance
- For both reasons SHA-3 is required to work well on FPGAs
- Programming requires circuit design knowledge
- The prices for development kits start at around 150€ plus tools
  - If you want to implement SHA-3 on an FPGA, it's quite an effort

Introduction to benchmarking & implementation of SHA-3

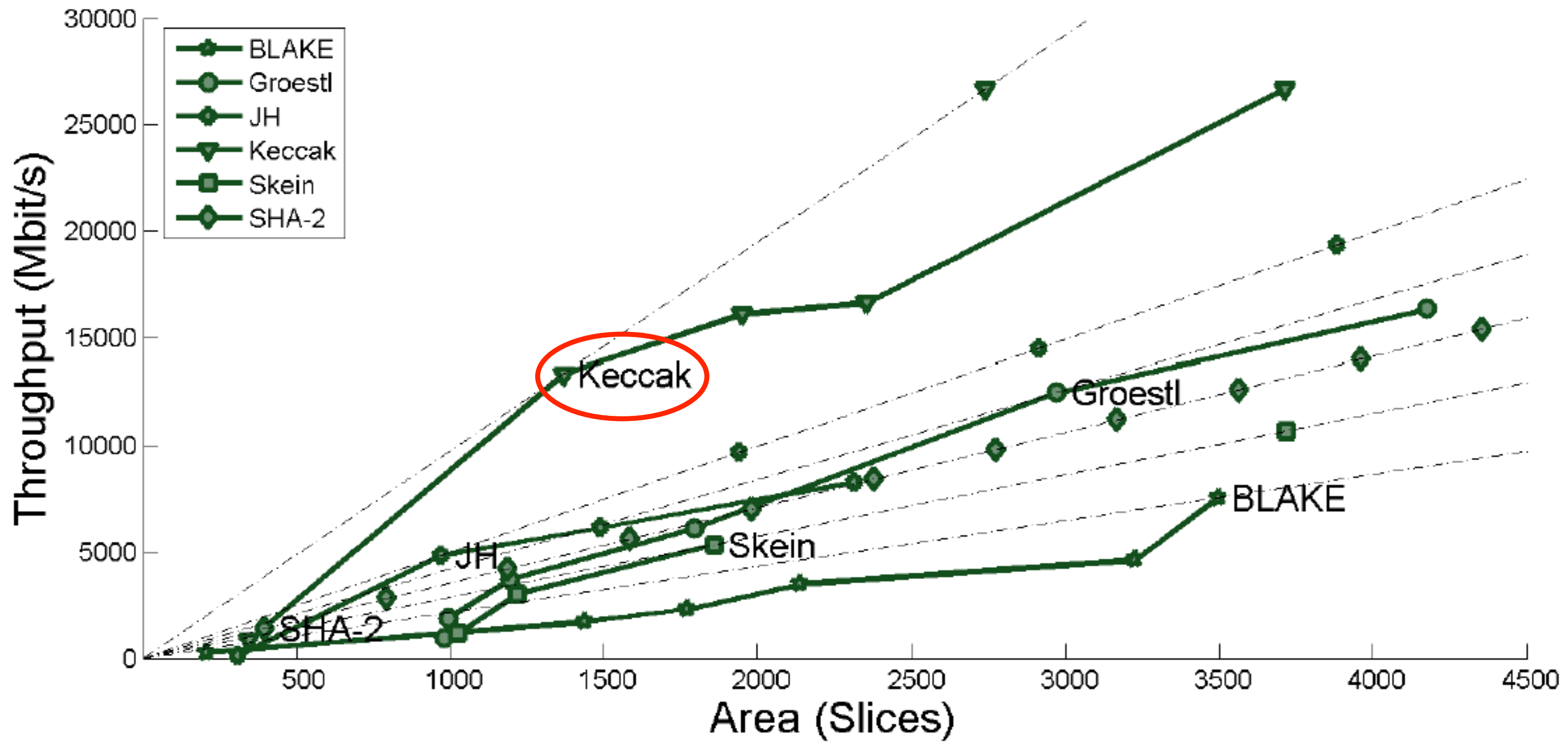
On FPGAs

- K. Gaj et. al., ATHENa project at George Mason University (GMU)
  - http://cryprography.gmu.edu/athena
  - Benchmarking, large results database, high and medium speed implementations
- B. Jungk at Hochschule Rhein-Main
  - http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012/documents/papers/JUNGK_paper.pdf
  - Low area implementations
- B. Baldwin, A. Byrne et. al. at University College Cork
  - http://www.ucc.ie/en/crypto/SHA-3Hardware/
  - Round 2 implementations

Introduction to benchmarking & implementation of SHA-3

On FPGAs

256-bit variants in Virtex 5

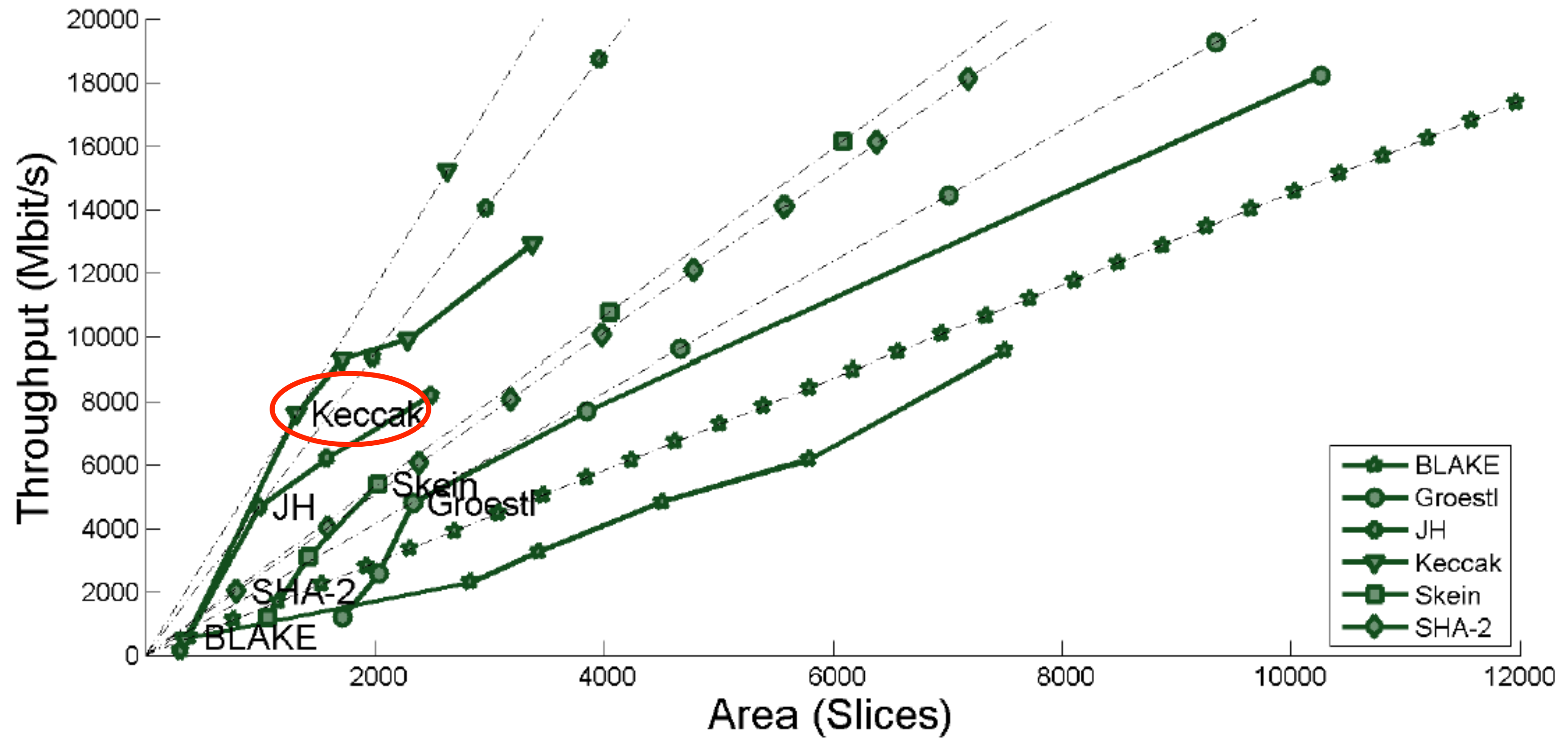


Source: Kris Gaj et. al., GMU: Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs

Introduction to benchmarking & implementation of SHA-3

On FPGAs

512-bit variants in Virtex 5

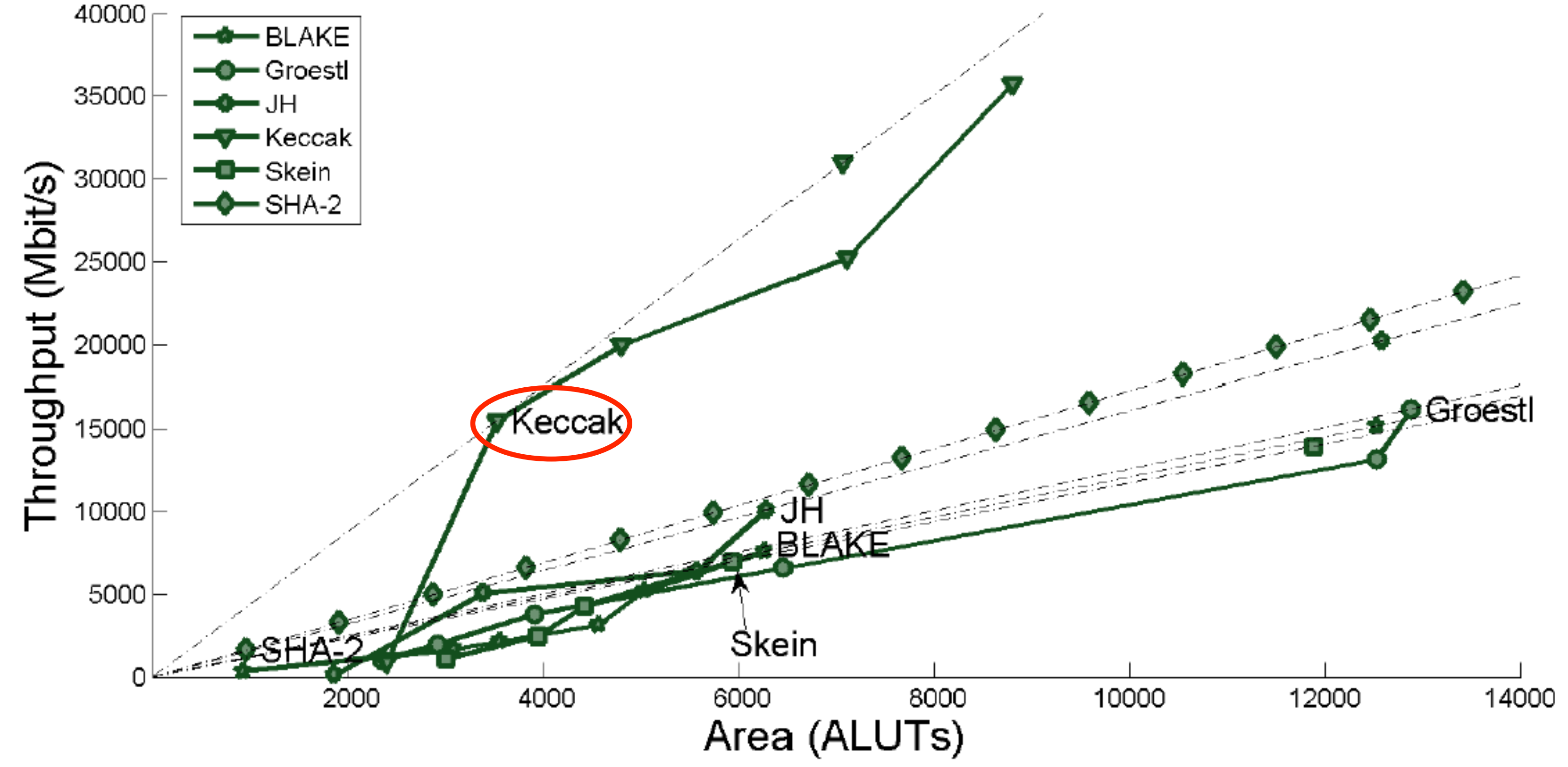


Source: Kris Gaj et. al., GMU: Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs

Introduction to benchmarking & implementation of SHA-3

On FPGAs

256-bit variants in Stratix III

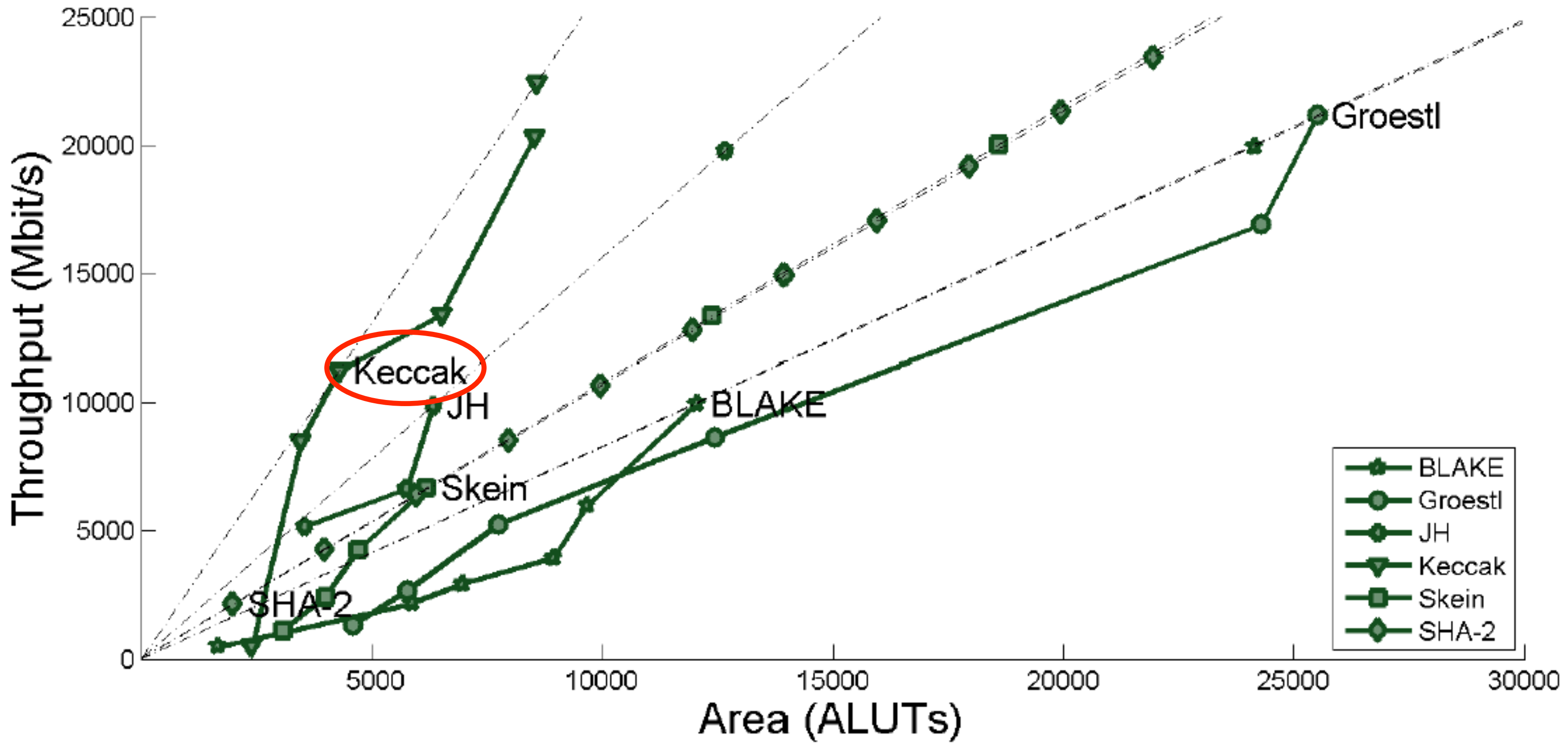


Source: Kris Gaj et. al., GMU: Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs

Introduction to benchmarking & implementation of SHA-3

On FPGAs

512-bit variants in Stratix III



Source: Kris Gaj et. al., GMU: Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs

Introduction to benchmarking & implementation of SHA-3

On ASICs - why?

- **A**pplication **S**pecific **I**ntegrated **C**ircuits (ASICs) are chips that are specifically designed for one application
  - There is no cheaper and more energy efficient way to perform that application – if you need a huge number of ASICs
  - Example: security chips for credit and debit cards
- It is very important that SHA-3 works well in ASIC implementations
- One can't "program" an ASIC, one needs sophisticated design tools and a full blown semiconductor fab to manufacture it
- There are no development kits
  - If you want to implement SHA-3 on an ASIC, you have to work at the right semiconductor manufacturer or at a very special research group

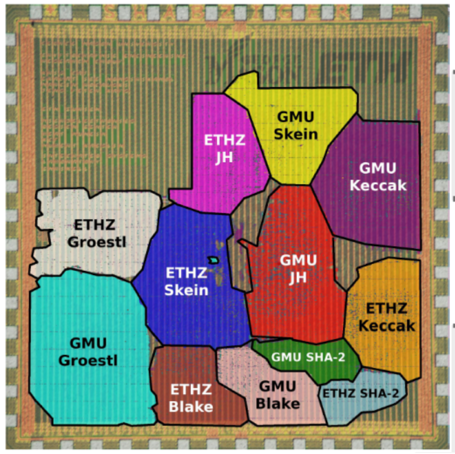
Introduction to benchmarking & implementation of SHA-3

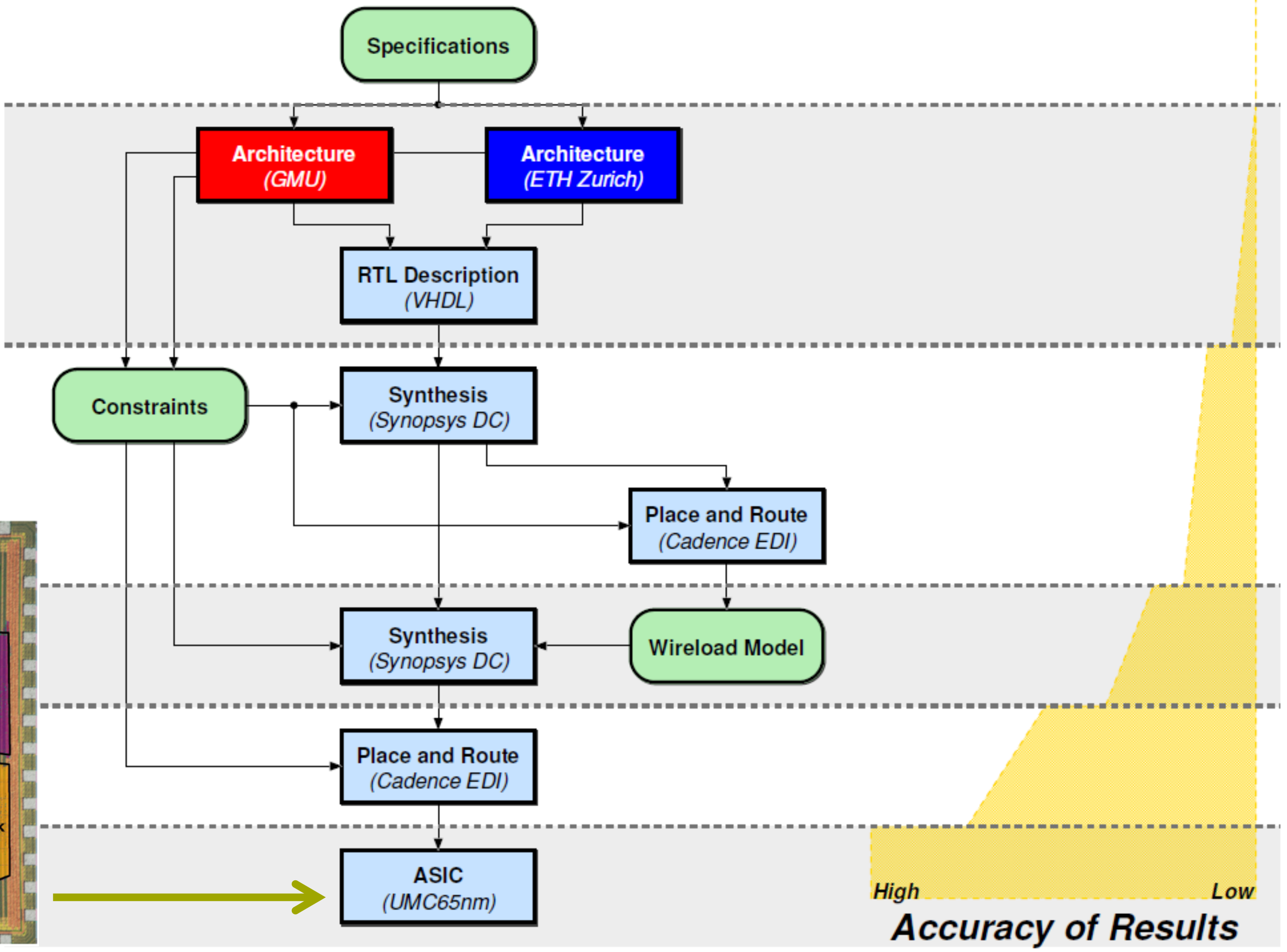
On ASICs

- P. Schaumont, L. Nazhandali et. al. at Virginia Tech
  - http://rijndael.ece.vt.edu/sha3/index.html
  - (Real 130nm ASIC) implementation and benchmarking
- F. Gürkaynak, K. Gaj, et. al., ETH Zürich / GMU cooperation
  - http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012/documents/papers/GURKAYNAK_paper.pdf
  - (Real 65nm ASIC) implementations (one from ETHZ and GMU each) and benchmarking

Introduction to benchmarking & implementation of SHA-3

On ASICs

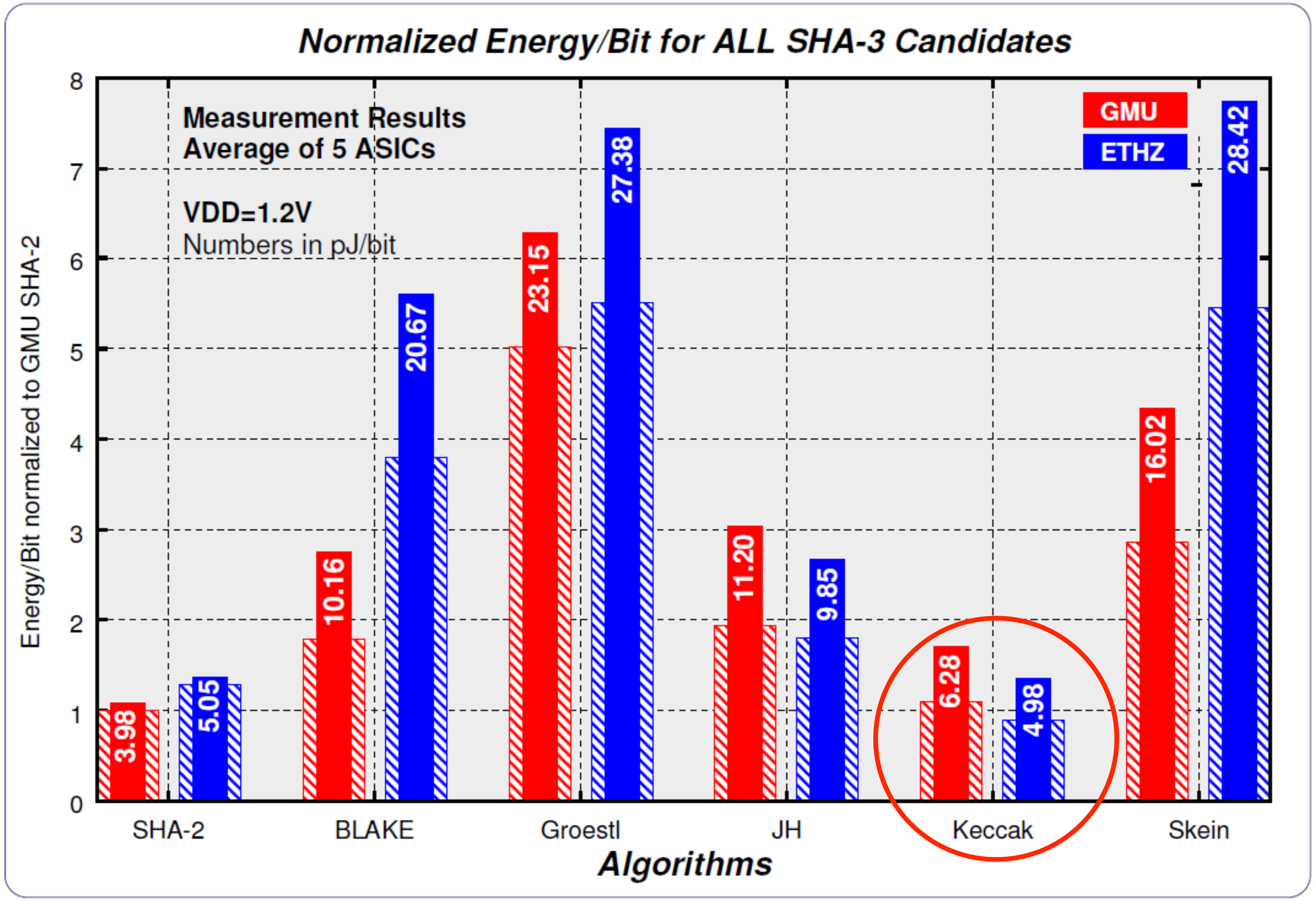




Source: F. Gürkaynak, K. Gaj, et. al., ETH Zürich / GMU: Lessons Learned from Designing a 65 nm ASIC for Third Round SHA-3 Candidates

Introduction to benchmarking & implementation of SHA-3

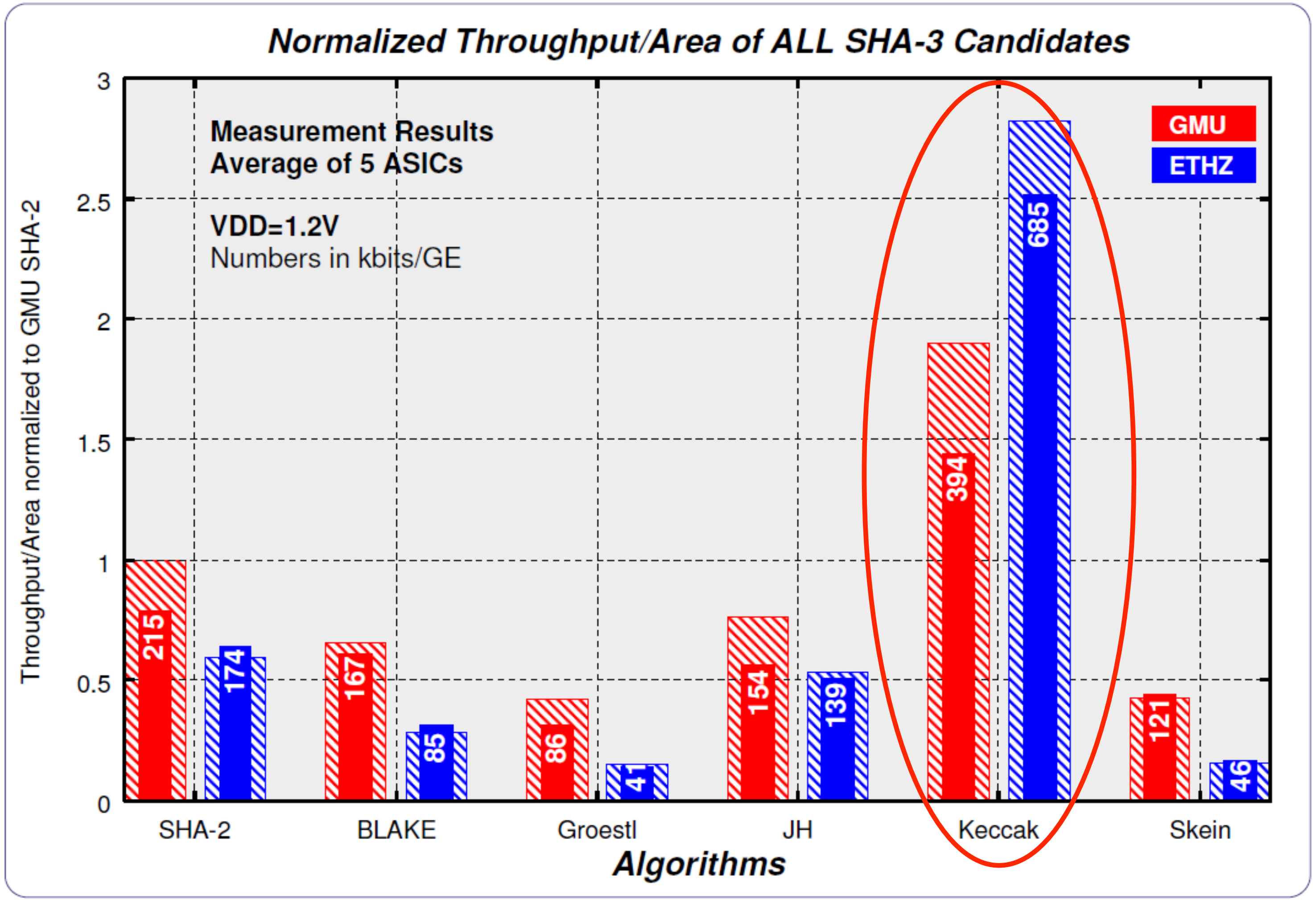
On ASICs



Source: F. Gürkaynak, K. Gaj, et. al., ETH Zürich / GMU: Lessons Learned from Designing a 65 nm ASIC for Third Round SHA-3 Candidates

Introduction to benchmarking & implementation of SHA-3

On ASICs



Source: F. Gürkaynak, K. Gaj, et. al., ETH Zürich / GMU: Lessons Learned from Designing a 65 nm ASIC for Third Round SHA-3 Candidates

SHA-3 embedded implementation and benchmarking in detail

How can implementations be optimized?

- For a specific purpose, e.g. speed or memory footprint
  - Speed: precompute as much as possible, unroll loops, use natural word length of target CPU even if too large for data, …
  - Memory: compute stuff on the fly, use smallest variable size possible, re-use subroutines and temporary memory (scratch pad)
- For a specific hardware
  - Build implementation from building blocks the target hardware provides
  - Need to know the target hardware very well
  - High degree of control over resource allocation necessary -> assembly
- Pareto always applies: go for inner loops / largest arrays first

Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012

SHA-3 embedded implementation and benchmarking in detail

Does benchmarking improve implementations?

- Yes!
- Keccak, 512 bit output, on 8-bit AVR microcontroller:
- August 2010, 2nd SHA-3 Conference:
  - 638 bytes of RAM, 3928 bytes of ROM, 7949 cpb, C implementation
- August 2010, 2nd SHA-3 Conference, 1 Day later:
  - 595 bytes of RAM, 3266 bytes of ROM (93,26% / 83,15%) by means of assembly implementations of parts of the code (Otte / Wenzel-Benner)
- March 2012, 3rd SHA-3 Conference:
  - 308 bytes of RAM, 1848 bytes of ROM, 1945 cpb, C/assembly implementation by Ronny Van Keer
- Implementers are heroes, too!

Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012

SHA-3 embedded implementation and benchmarking in detail

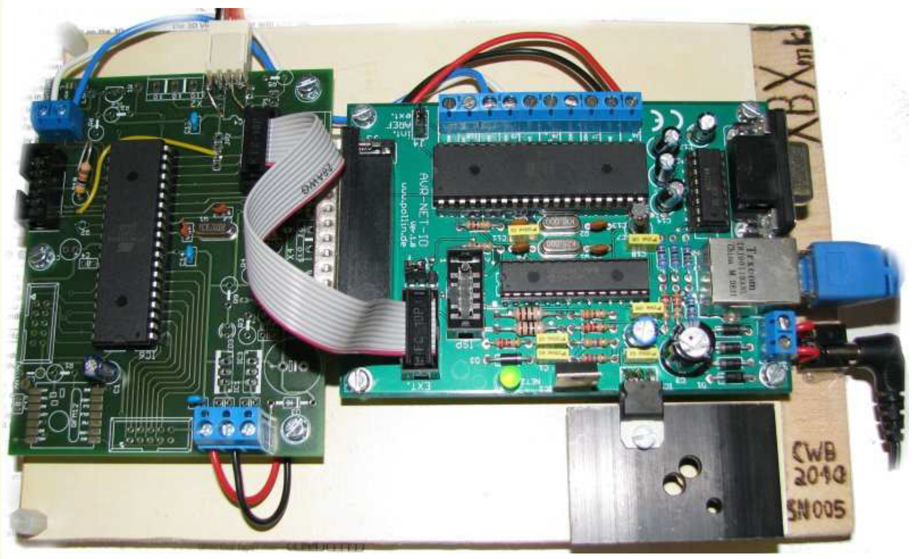
What is XBX?

- XBX: Benchmarking of ’small devices’ that
  - can execute compiled C code
  - can’t run a POSIX compliant operating system
  - can’t run a C compiler
  - are often embedded in consumer electronics

SHA-3 embedded implementation and benchmarking in detail

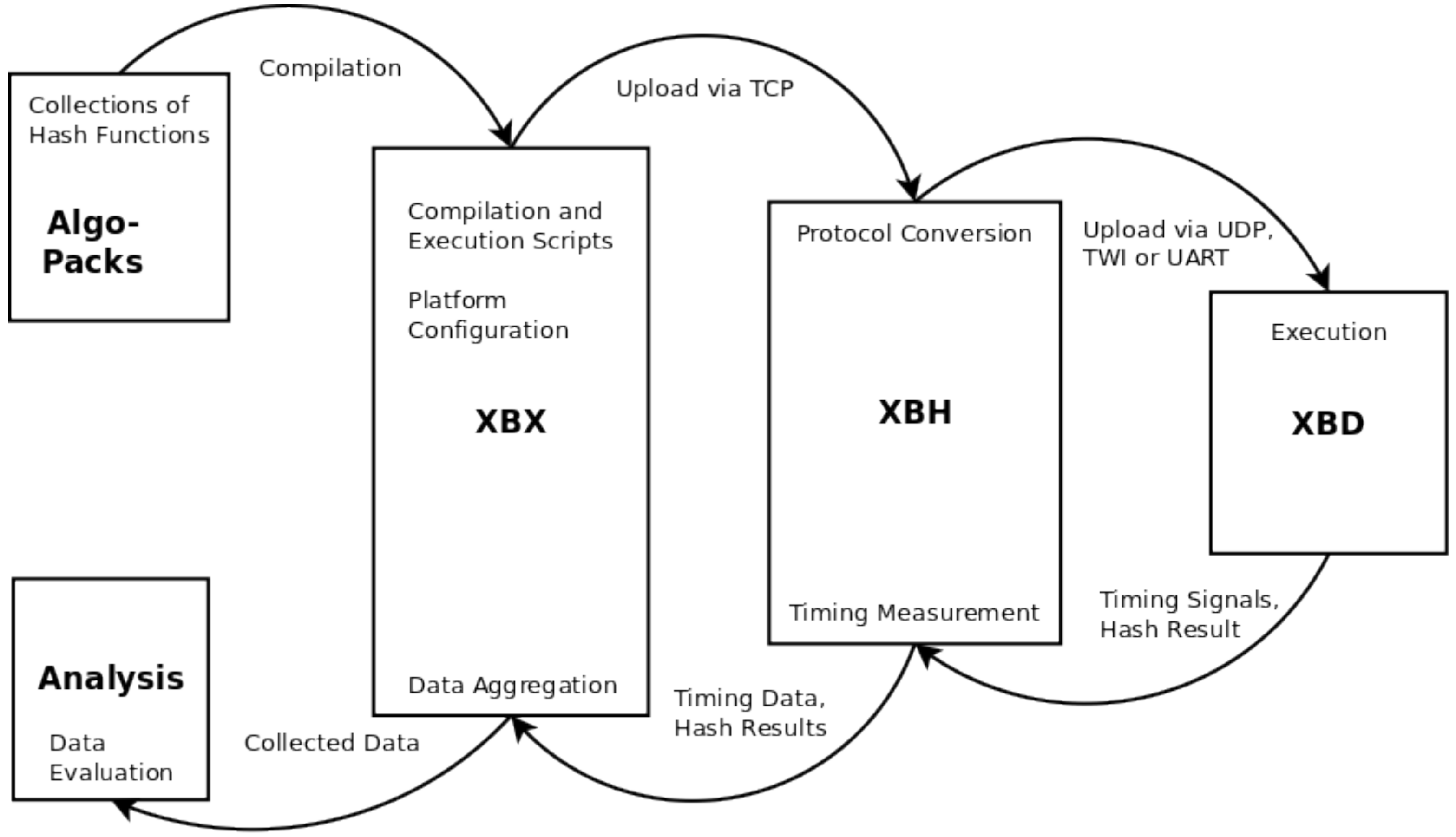
Why XBX?

- Small devices require a different approach to benchmarking:
  - Binaries have to be created on another system
  - Memory footprint is an important metric
  - Standardized timing services are unavailable



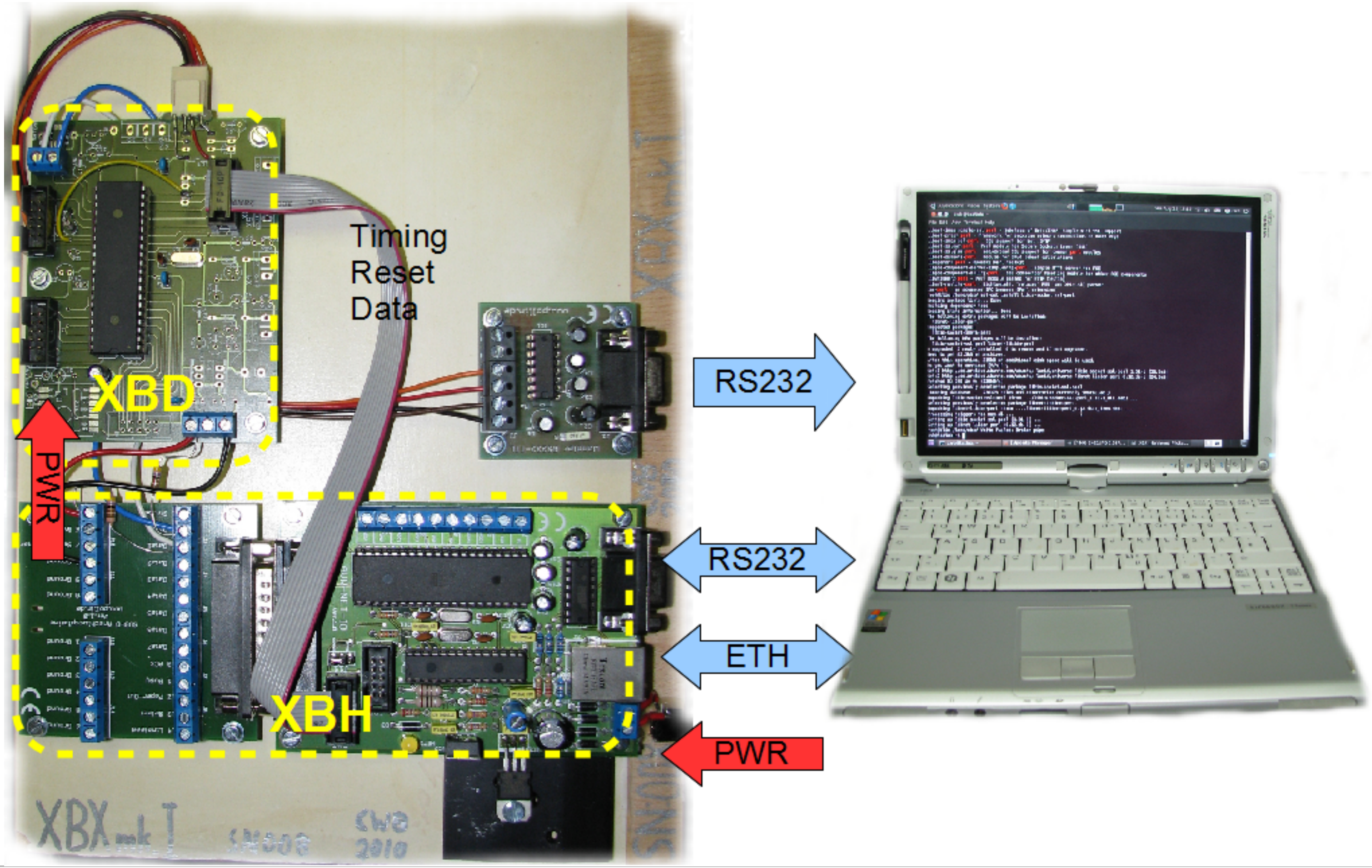
SHA-3 embedded implementation and benchmarking in detail

How does XBX work?



SHA-3 embedded implementation and benchmarking in detail

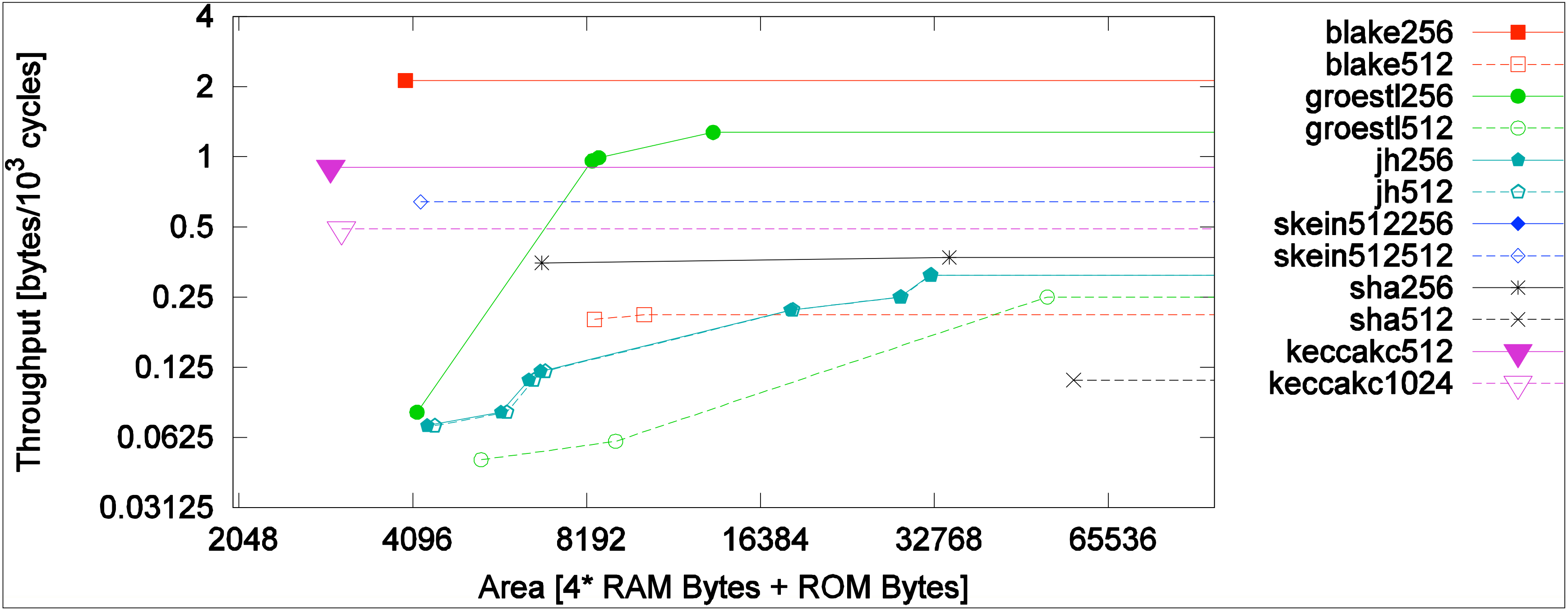
How does XBX work?



SHA-3 embedded implementation and benchmarking in detail

AVR (8-bit): Atmel ATmega1284P

- Best XBX platform for estimating smart card performance
- Memory footprint most important, focus on 256-bit hashes
- Keccak looks really good

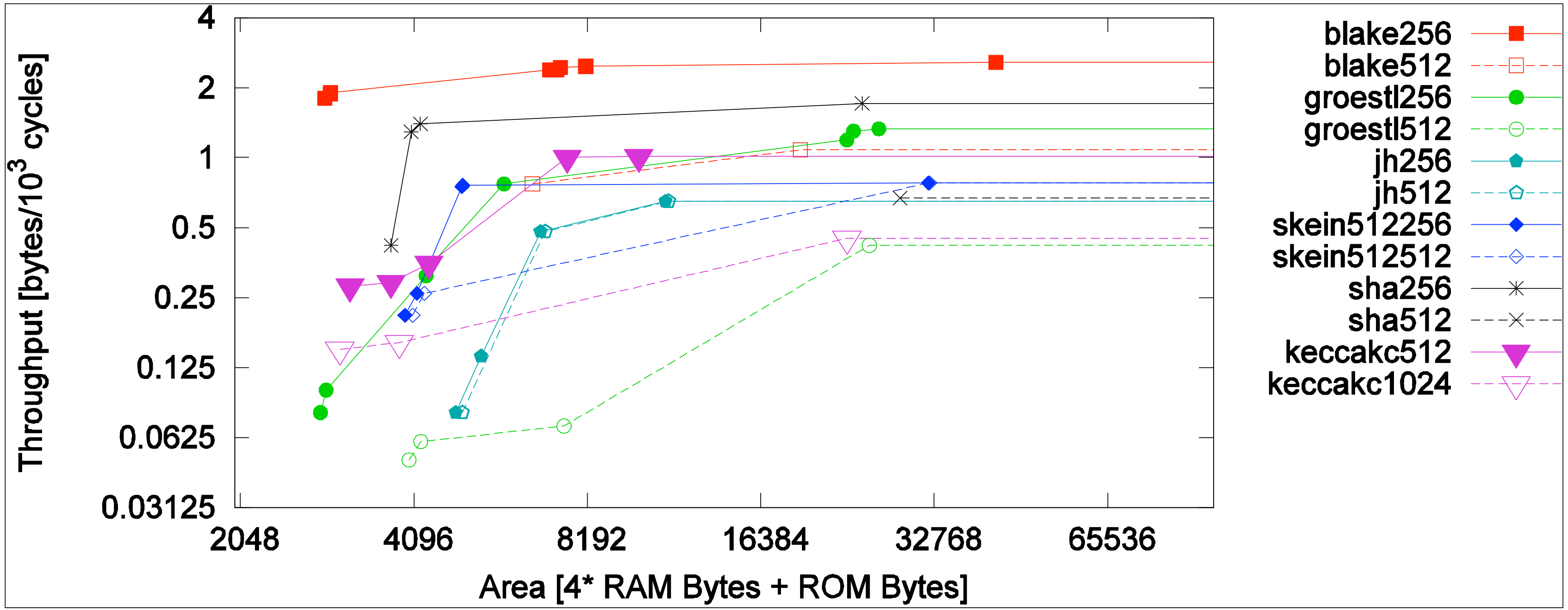


Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012

SHA-3 embedded implementation and benchmarking in detail

MSP430 (16-bit): Texas Instruments MSP430FG4618

- Low power platform, setup developed at GMU
- Memory footprint most important, focus on 256-bit hashes
- Keccak does OK

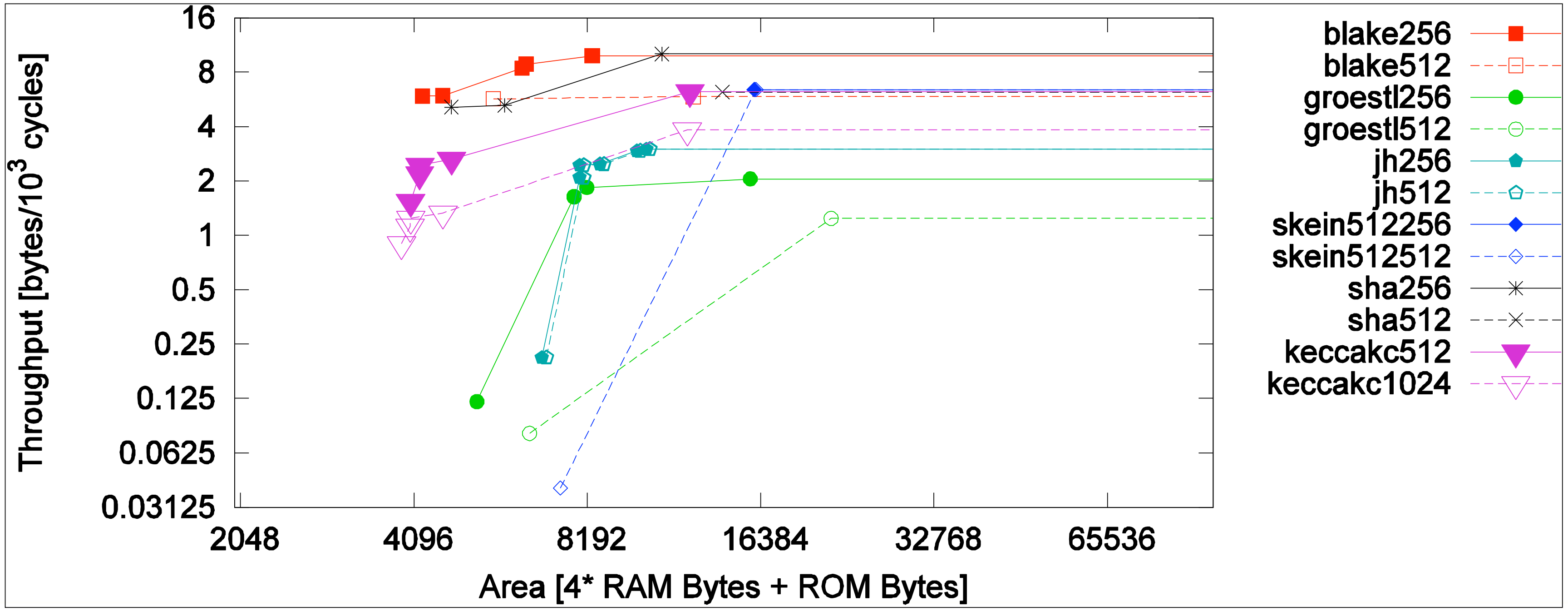


Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012

SHA-3 embedded implementation and benchmarking in detail

MIPS (32-bit): Texas Instruments AR7

- MIPS core, Linux based, popular in DSL routers
- Throughput most important, no output length focus
- Keccak does OK

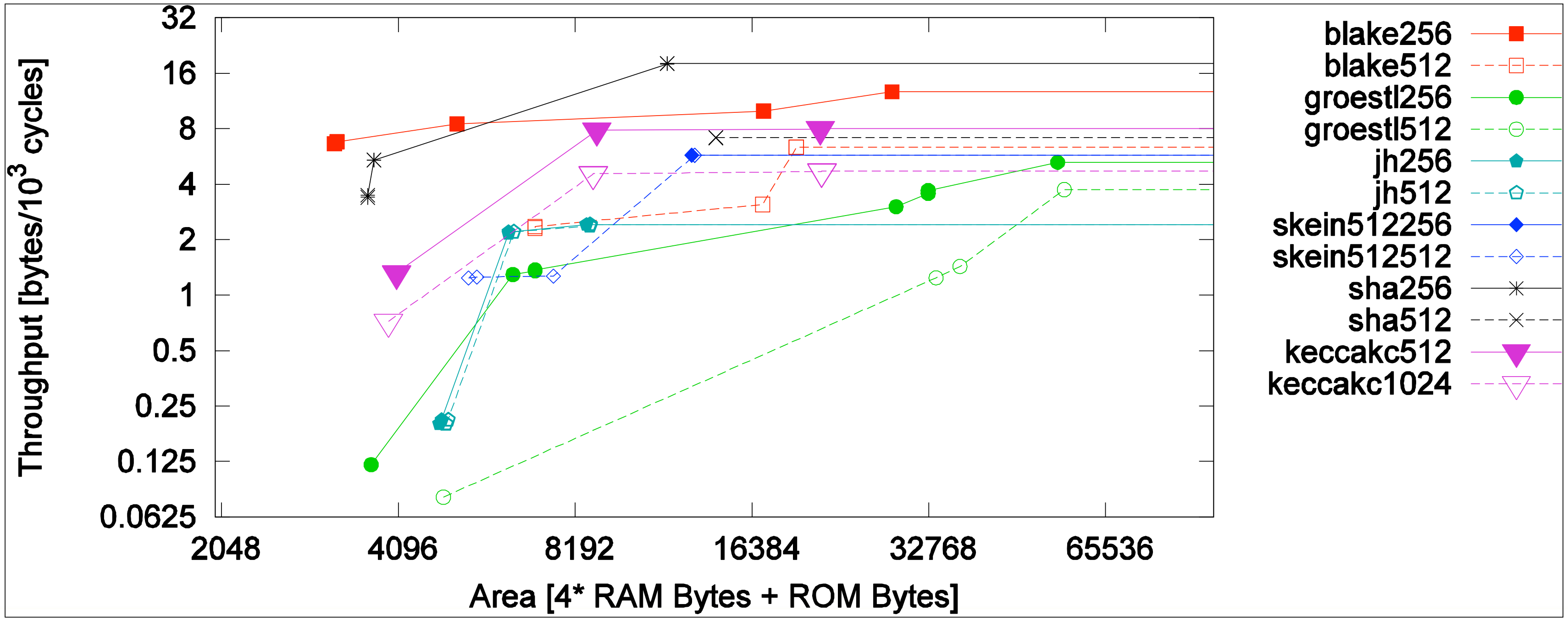


Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012

SHA-3 embedded implementation and benchmarking in detail

ARM 920T (32-bit): Atmel AT91RM9200

- Older ARM core, Linux based, popular in automation
- Throughput most important, no output length focus
- Keccak does pretty well

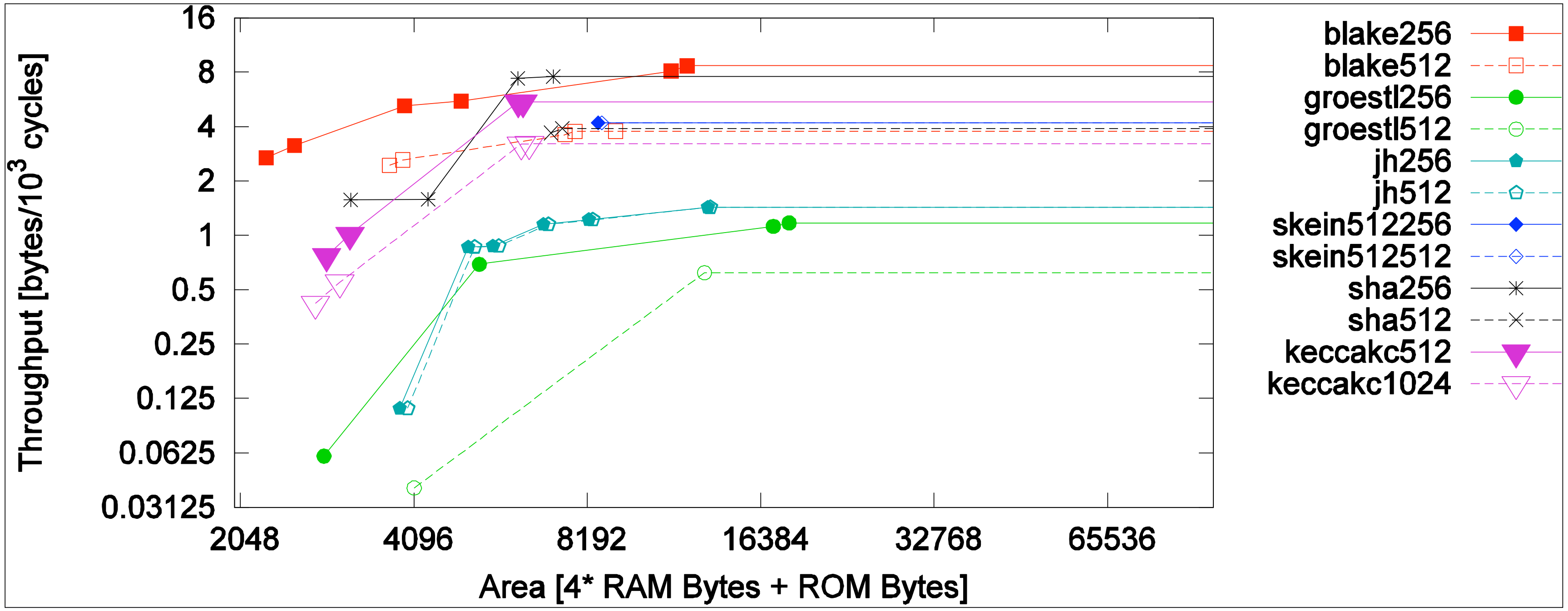


Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012

SHA-3 embedded implementation and benchmarking in detail

ARM Cortex-M0 (32-bit): NXP LPC1114

- Current ARM core, low cost, used in microcontrollers
- Memory footprint most important but no output length focus
- Keccak does pretty well

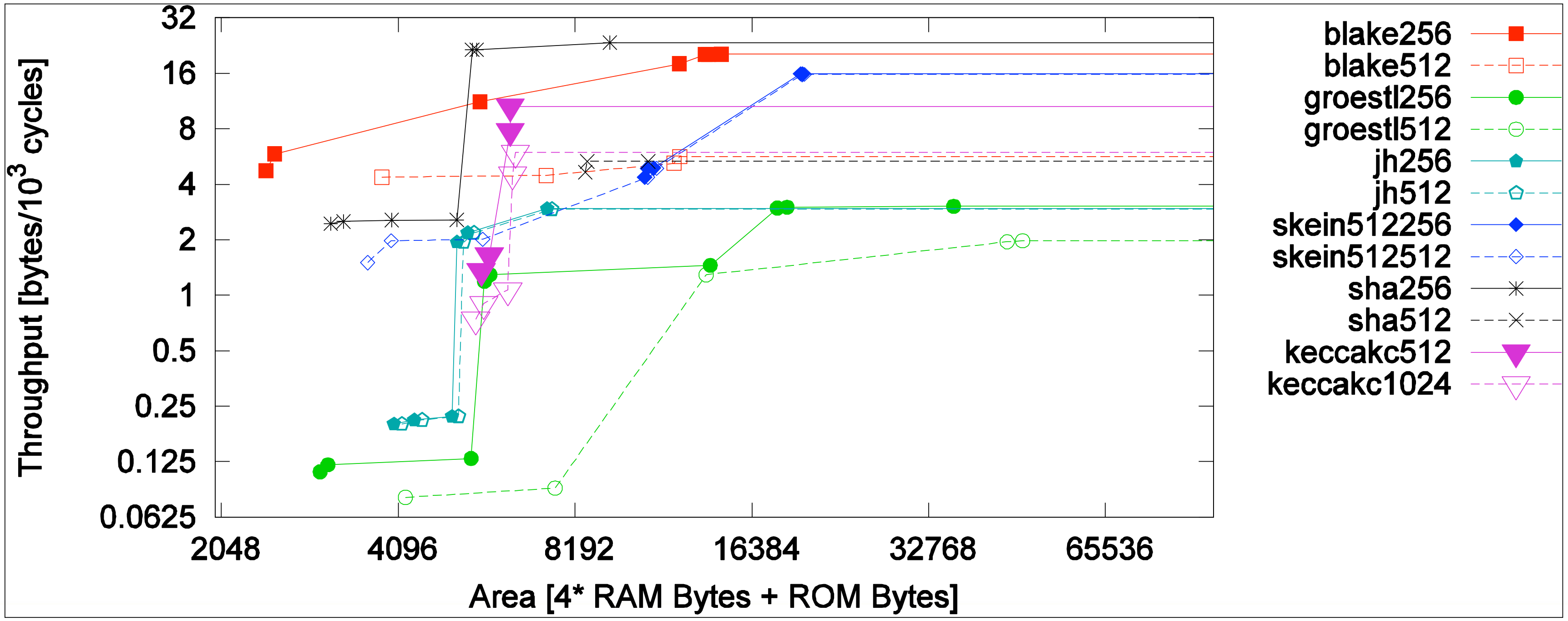


Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012

SHA-3 embedded implementation and benchmarking in detail

ARM Cortex-M3 (32-bit): Texas Instruments LM3S811

- Current ARM core, cost-performance balanced, two criteria
- Low memory footprint and speed
- Keccak does OK, but not great

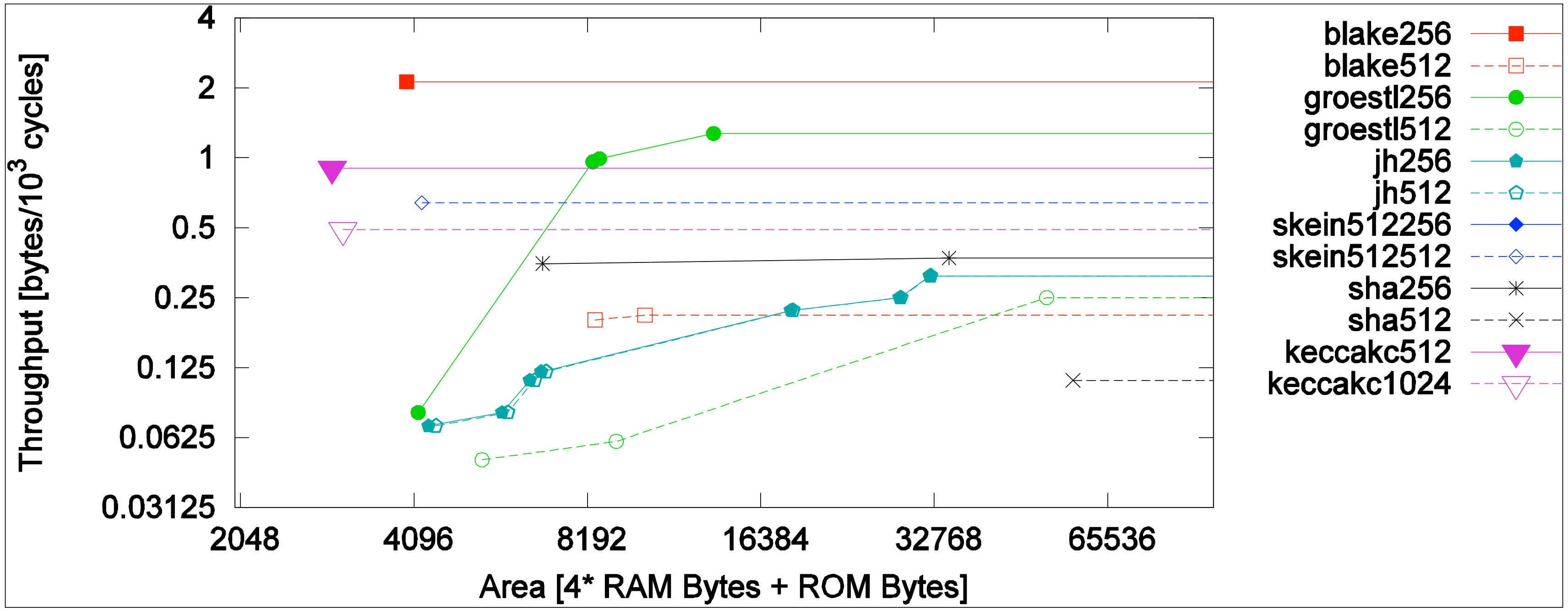


Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012

SHA-3 embedded implementation and benchmarking in detail

ARM Cortex-A8 (32-bit + SIMD): TI DM3730

- Current ARM core with vector extensions, Linux based
- Throughput most important, no output length focus
- Keccak does OK, but Skein rules



Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012

SHA-3 embedded implementation and benchmarking in detail

Questions?

Source: C. Wenzel-Benner, J.Gräf et al.: XBX Benchmarking Results January 2012