Galois Ultra Low Power High Assurance Asynchronous Crypto

Joe Kiniry
October 2016

Presented on behalf of Galois,
the University of Southern California (Prof. Peter Beerel), and
Reduced Energy Microsystems (William Koven)

Executive Summary

- R&D performed with USC and REM has two main goals:
  - *synthesize high assurance HDL from formal specifications*, and
  - experiment with *asynchronous VLSI* crypto
- core strategy is to:
  - start with formally verified Cryptol specifications of three lightweight crypto algorithms (Simon, Speck, and AES),
  - automatically synthesize System Verilog-CSP (SVC) implementations and assurance artifacts,
  - fabricate test chips in a fairly old process (IBM 130 nm), and
  - characterize correctness, performance, and energy

Core Results

- all Simon and Speck cores operate correctly; multi-stage AES and the hardware error counter do not (there were design errors in hardware CAD; the HDL is correct)

- all but one chip operate properly (fair yield)

- measured frequency estimate (as if clocked) varies from ~300 MHz@~0.6 V to 2.1 GHz@1.35 V

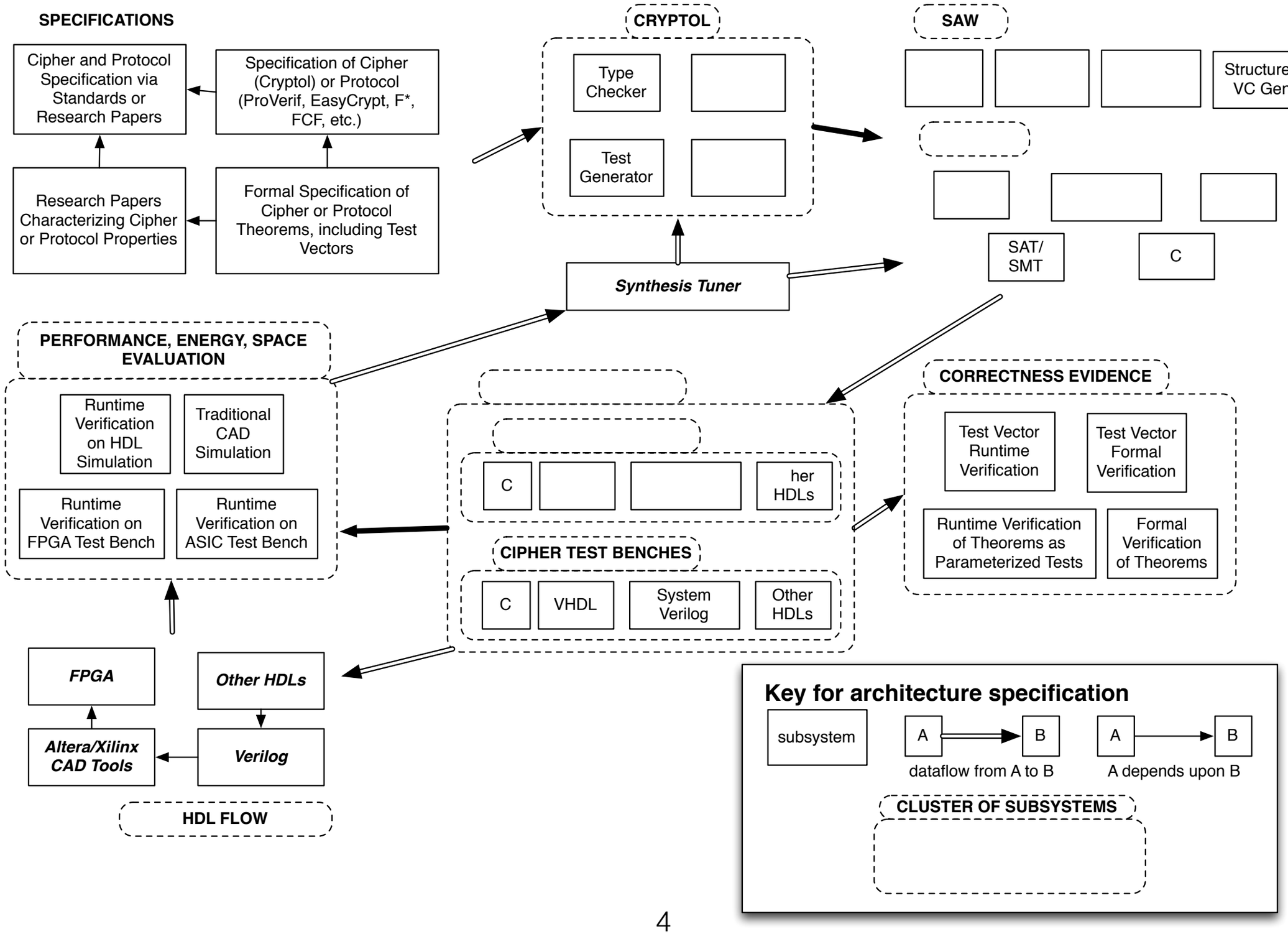
- energy/bit at 140 nm ranges between 4.2 and 7.5 pJ/b at threshold

- static load varies between 0.0012 mA and 0.0029 mA

- energy varies between 0.05 mW and 10 mW and is in close alignment with (post place and route) simulation

- safe minimum operating voltage is the process threshold (~625mV)

Galois HACrypto Toolchain



Hardware Synthesis

- we have synthesized formally verified high performance System Verilog-CSP implementations of Simon, Speck, and AES

- the hardware synthesis pipeline is

  Cryptol → Abstract Circuit Representation → System Verilog-CSP

- Cryptol modules are mapped to CSP processes and function calls are mapped to CSP message sends

- our focus is on pipeline simplicity (for assurance) at this time; no pipeline optimizations were made to achieve high performance or low energy

- there are a variety of research opportunities with respect to secure asynchronous VLSI and platform targeting

Estimating Performance and Energy for Later Processes

- we fabricated using IBM's 130 nm process
- our implementations, broadly speaking, use 4.2–7.5 pJ/bit at our lowest power (0.625 mA)
- a 40% reduction in area/power for each generation is generally a good estimate, but there are caveats
- other work discussed later focuses on 65 nm simulation
- consequently, we estimate that our energy use in 65 nm is 1.5 pJ/bit, and in the fJ/bit in the latest processes

Estimate Caveats

- 130 nm library we used had cells with only a single channel length and only a single Vt (threshold); by 65 nm and beyond, most libraries have cells with the same logic function (and size) but different channel lengths and different Vt's to allow for power/ performance optimizations that didn't exist in our process

- there is also a bigger reduction in power/area from something like 90 nm to 55 LP (a optical low power shrink of 65 nm offered by both Global Foundries and TSMC) and similarly a bigger jump from 65 nm to 40 LP (again an optical low power shrink of 45 nm)

- so by the time you get all the way to something like 28 LPP, you almost get another generation's worth of improvement than would be implied by 130 -> 28 LPP via just 40% per generation

Software Synthesis

- we have also *automatically synthesized formally verified high performance software implementations* of all three ciphers

- this synthesis pipeline transforms Cryptol programs into their SAW IR representations via symbolic evaluation, and then transforms that representation directly into LLVM

- our focus in this pipeline is on simplicity (for assurance) and where the opportunities for improvement are (for security)

- there are obvious R&D opportunities wrt formally verified side channels (e.g., Almeida et al.) and platform targeting (via automatic evaluation of functional and non-functional properties)

Assurance

- assurance means providing third-party verifiable evidence that claims we make are true (in all circumstances, given any input, etc.)
    - the strength of an assurance argument ranges from “we did some code review and ran a few unit tests” to “we formally specified and verified the following properties”
- our assurance case is based upon *Literate Cryptol* specifications
    - specs are literate (in the Knuth sense) versions of NIST and IETF standards
    - specs include models, reference implementation, and many theorems
    - spec compile to, e.g., NIST PDFs and ASCII RFCs
    - specs are also interpretable as mechanized models of algorithms and protocols, thus theorems are automatically proven (about models and implementations) and/or are used to automatically generate test benches

Formal Verification

- formal verification is about proving theorems, sometimes automatically and sometimes interactively
- our theorems focus on correctness; others are about security
- we automatically prove theorems about *specifications* and *relationships among specifications and implementations*
- some example theorems include
  - a decrypt of an encrypt is what we started with
  - this optimized code behaves exactly as that reference code
  - this LLVM compiled from that C behaves exactly as specified in that Cryptol specification for all possible inputs

Formal Validation

- formal verification is only possible with a mechanized semantics
- most hardware engineers do not understand proof, though some do understand Jasper-style equivalence checking
- we synthesize complete test benches from specifications by transmuting all theorems into SVC test code and assertions
- test benches are checked using a variety of techniques available in modern CAD tools (mainly simulation, finite explicit state model checking, and equivalence checking)
- with additional resources we could write a full mechanized semantics of System Verilog-CSP and provide even greater assurance

Related Work

- comparisons to the state of the art are difficult
- other implementations are clocked, report rough energy estimates from simulations rather than measurements, and often optimize for size

- in a clocked setting, size^2 ~ energy, but in an unclocked setting, there is little relationship

Big Picture Results

	Block speed (KHz)		Bit speed (kbps)	Power (mW)		Energy/bit (pJ)	
	Chip	Simulation	Chip (Sim for AES)	Chip	Simulation	Chip	Simulation
simon128	1,916.80	1,958.15	250,643	6.826	7.289	27.82	29.08
simon48	3,788.40	4,011.18	192,537	4.149	4.449	22.81	23.11
speck128	3,841.90	4,088.88	523,377	10.204	11.300	20.75	21.59
speck48	6,001.20	6,231.51	299,112	4.489	4.862	15.58	16.25
aes128		28,737.30	3,678,374		659.400		179.26
Results at 1.2 V, using TT library for simulation, all delay lines set to minimum viable setting							
Simulation results using "SigCMin" SPEF with "MINIMUM" SDF values							

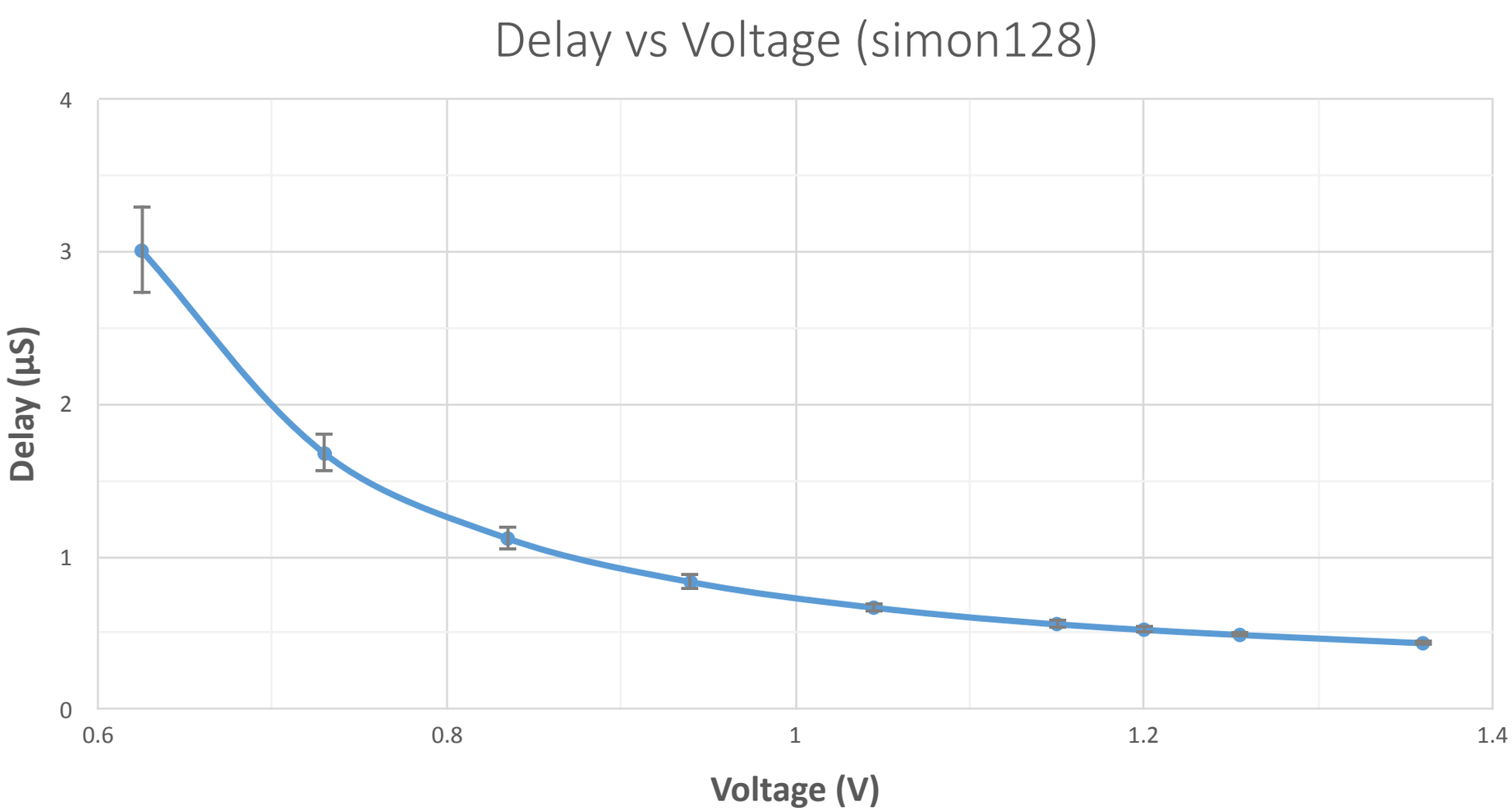
Performance

- reported maximum performance implementations in FPGA are in the 2.5 to 5.3 Mbps range (Gulcan, Aysu, Schaumont)

- reported maximum performance of other lightweight ciphers (not Simon & Speck) at 1.2 V & 65 nm process is 2 to 15 Gbps (Kerckhoff et al.)

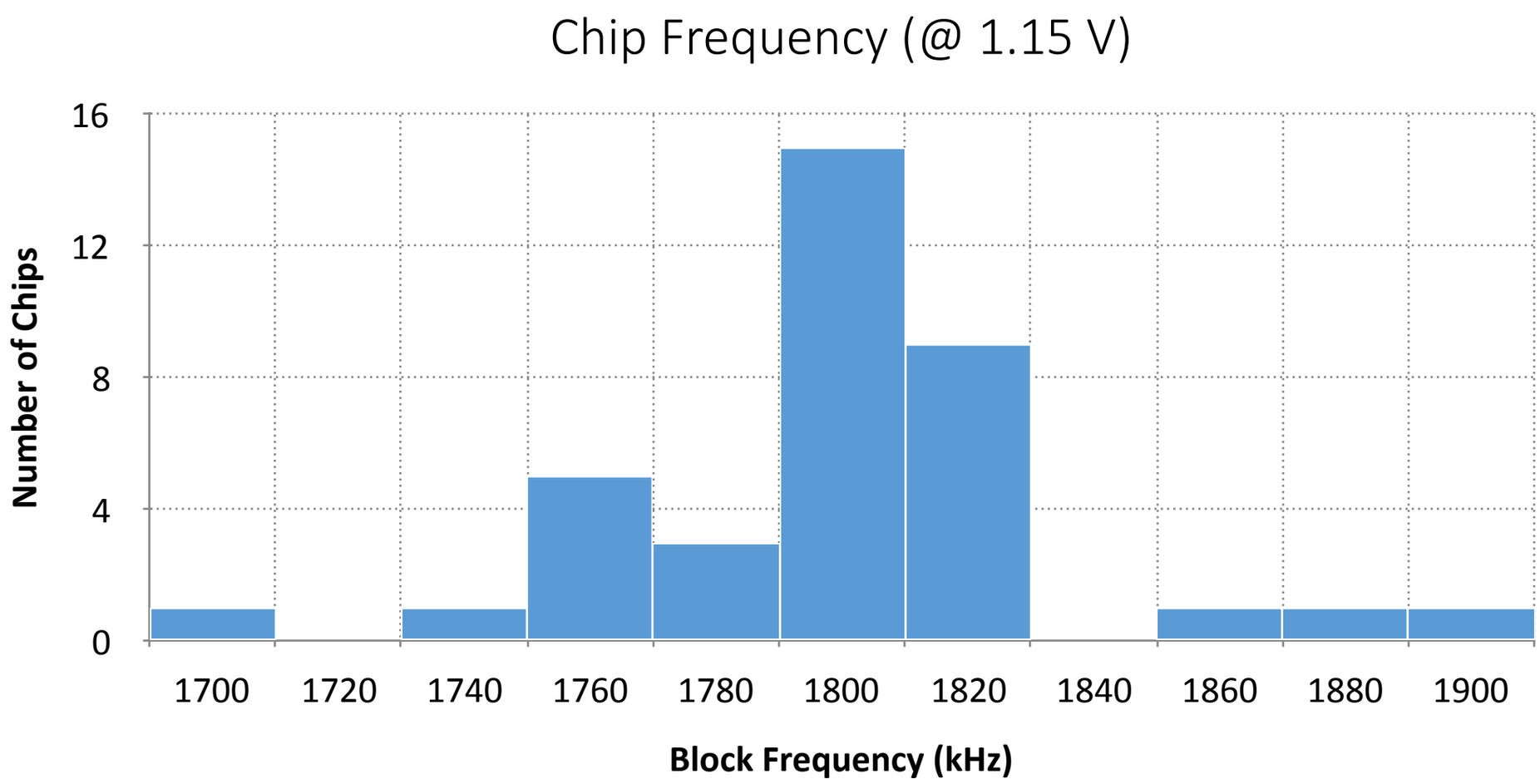
- our (unoptimized) performance ranges from 170 Mbps (Simon 48) to 450 Mbps (Speck 128)

Simon 128/128 Delay



- data collected from a single chip
- varied voltage from 0.538 V to 1.46 V
- shows internal cycle time of the core

Chip-to-Chip Variation

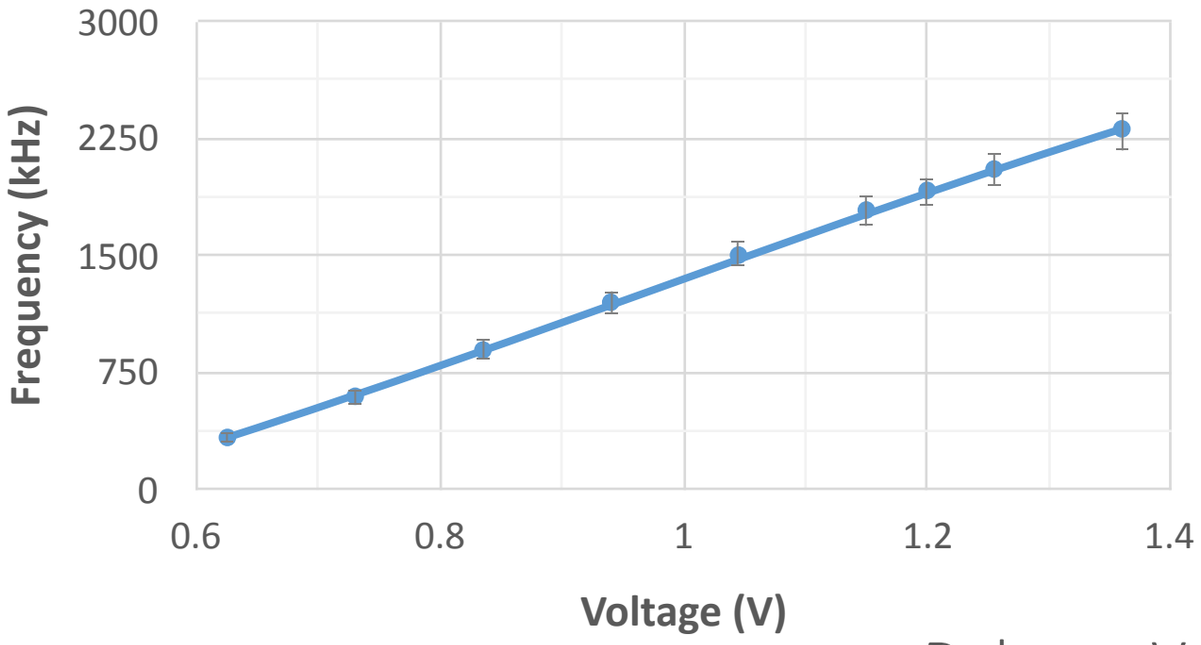


- tested 38 chips; 1 failed
- showing frequency of simon128 core @ 1.15 V (internal frequency is 66x faster)

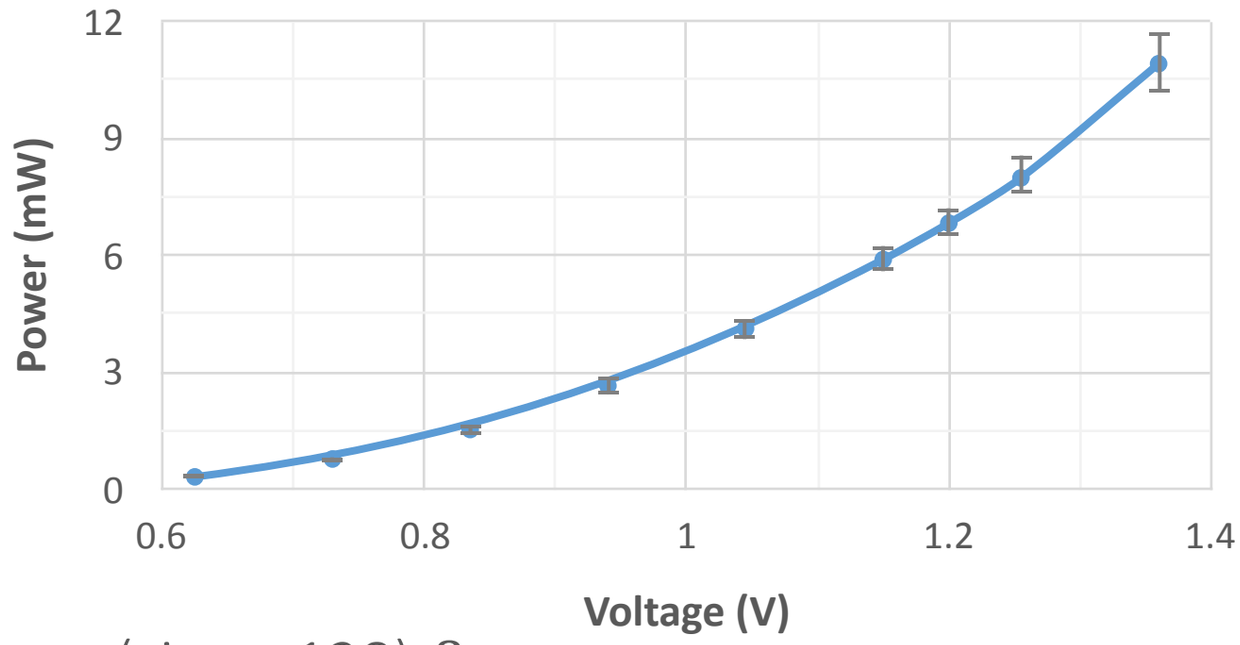
Energy

- we run correctly at threshold voltage
- we are power invariant and performance scales nicely
- our energy measurements are for the encryption cells only; all the I/O pads, logic, and memory used for testing on a separate power domain
- precise numbers for pJ/bit for a few algorithms follow
- energy use in low-power scenarios is a composite of active and quiescent energy, thus our low quiescent energy (1 μA) is exciting
- we extrapolate for modern processes in the following slides

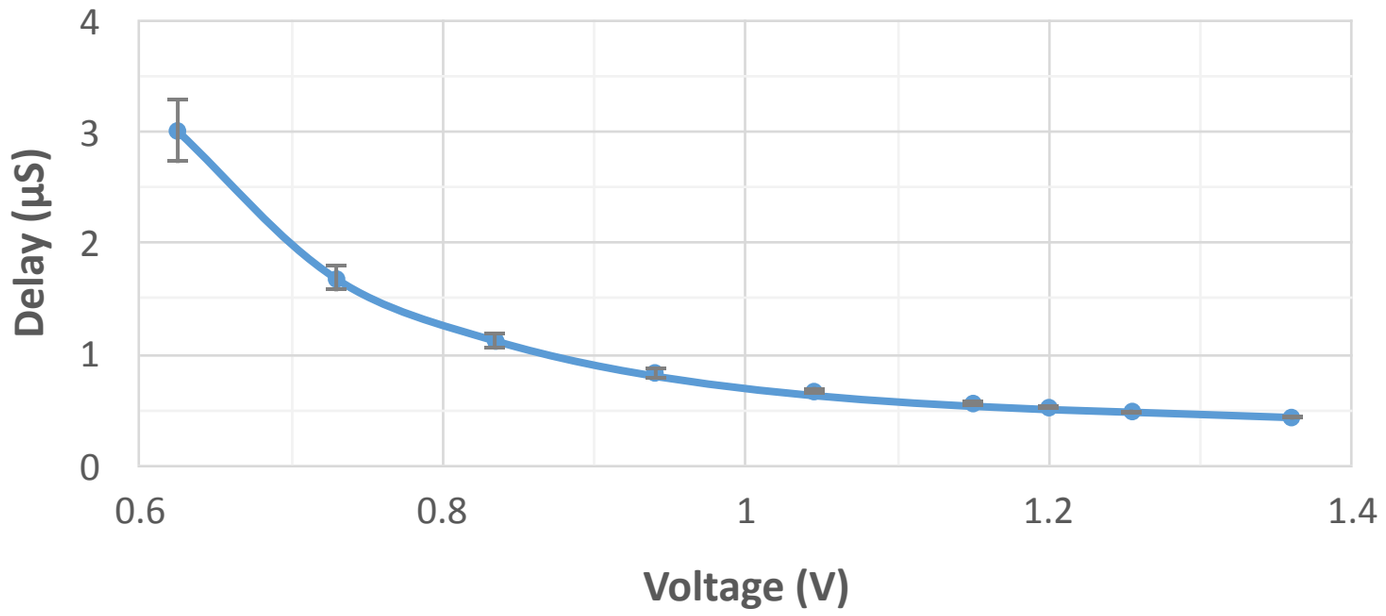
Frequency vs Voltage (simon128)



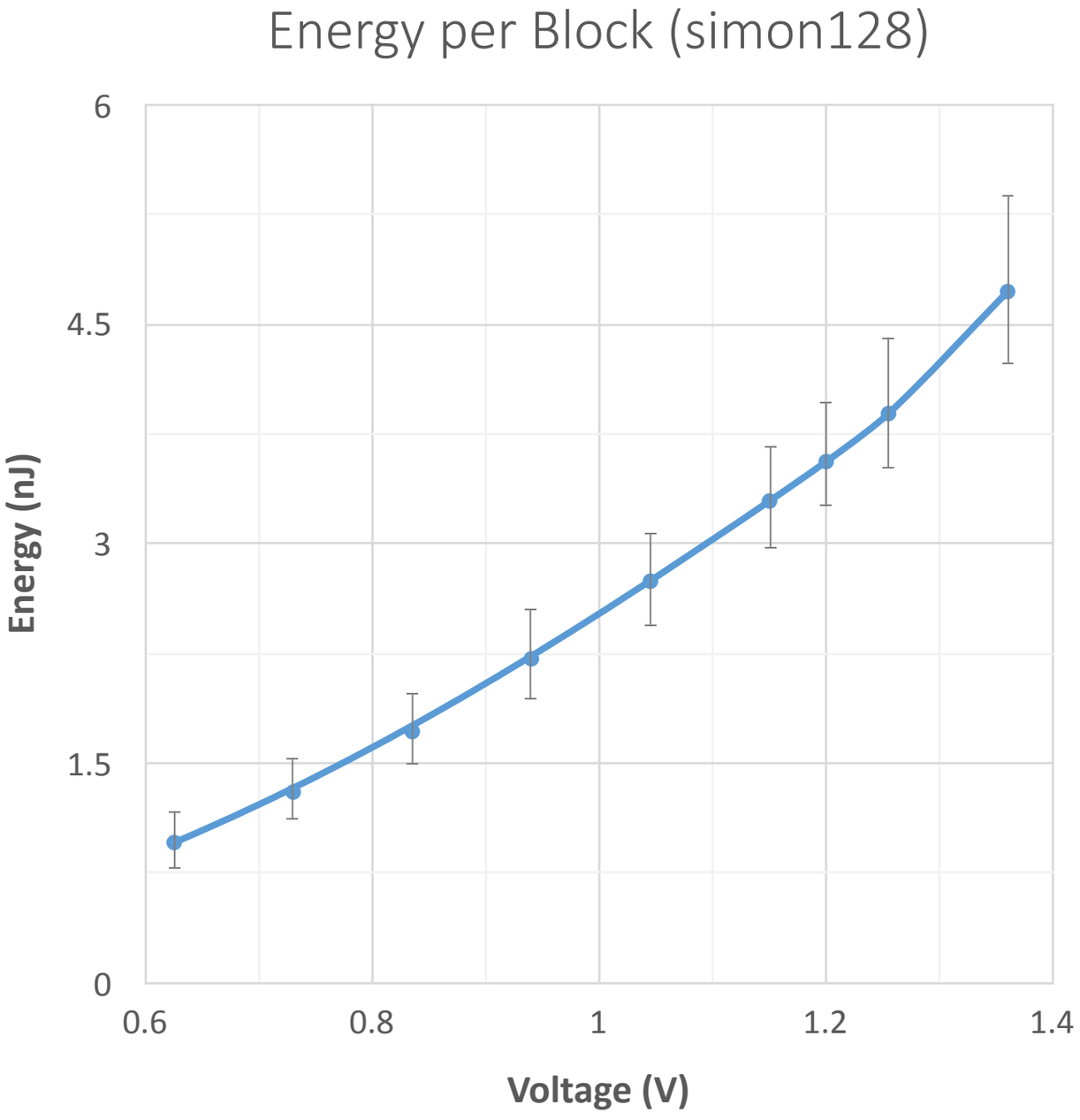
Power vs Voltage (simon128) &

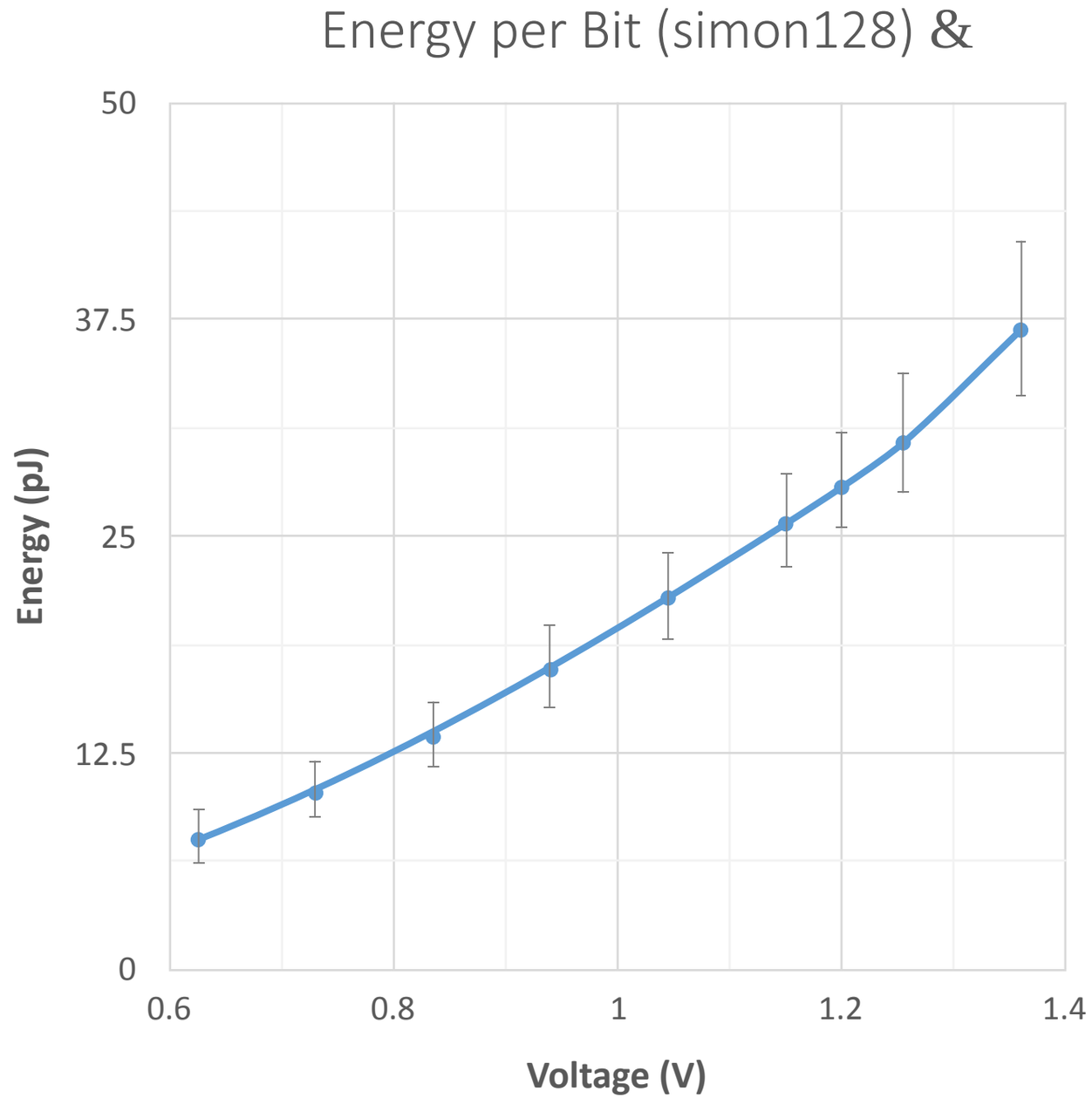


Delay vs Voltage (simon128) &

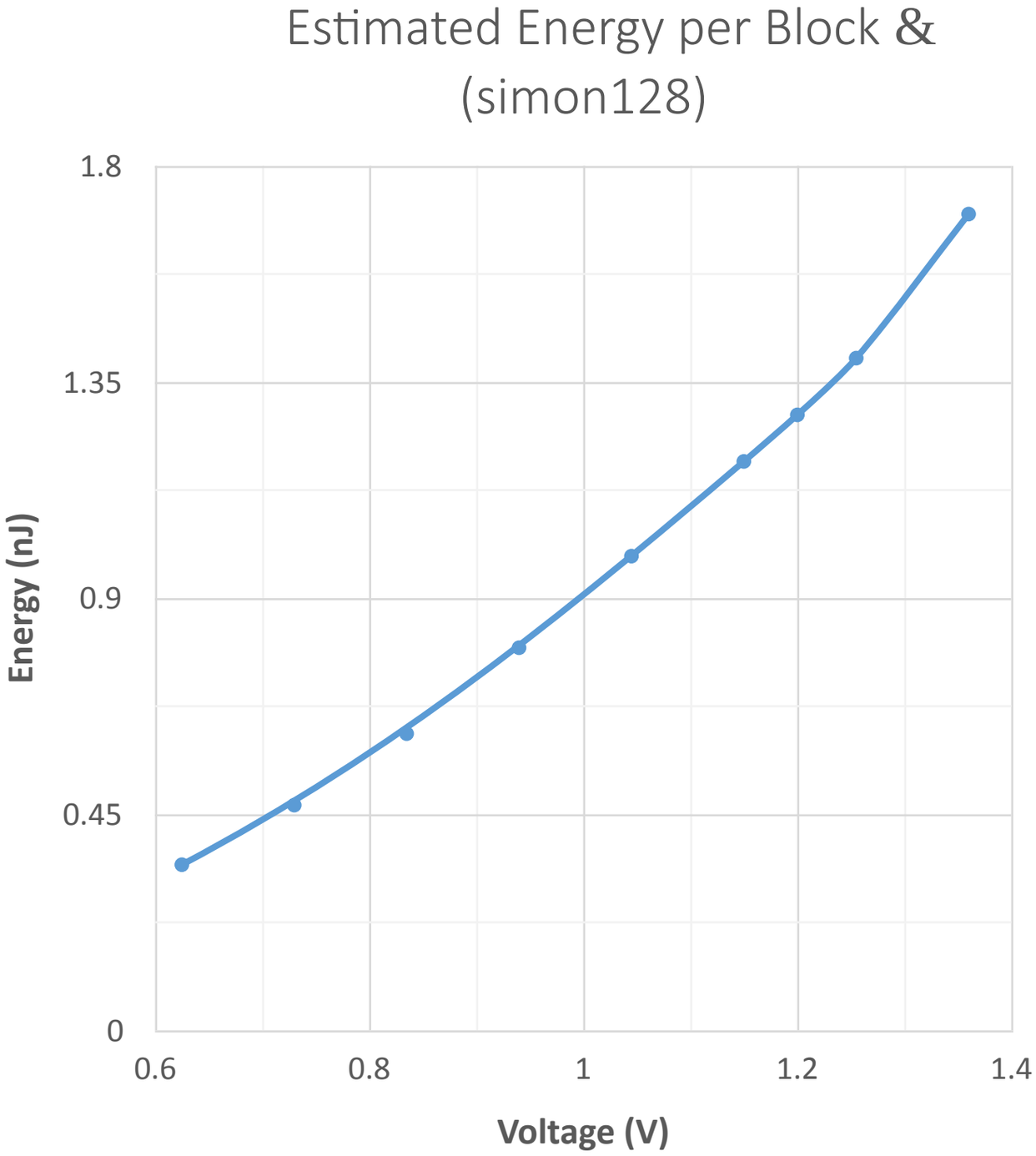


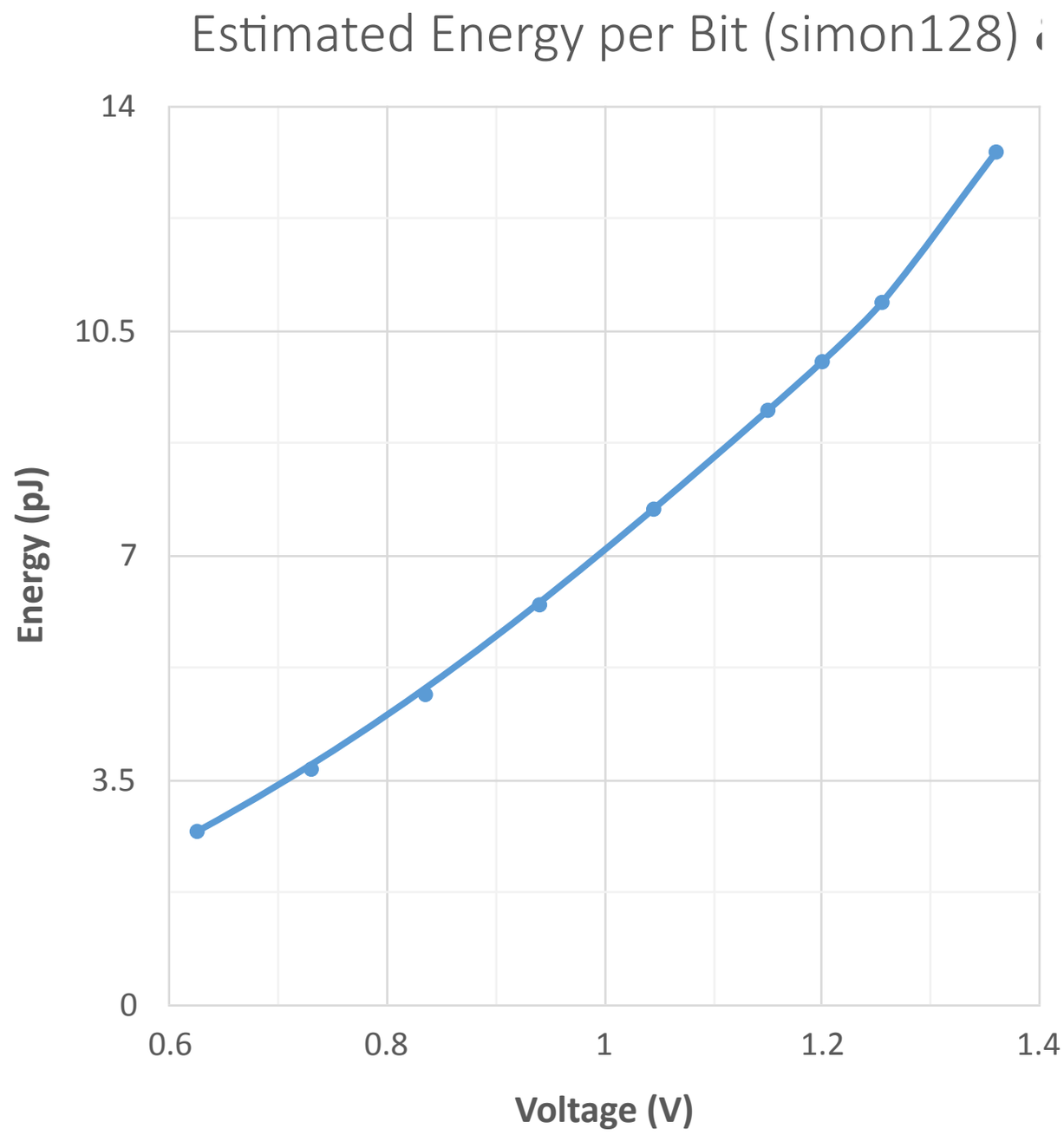
- averaged results from entire lot of chips (130 nm)
- fastest / average / slowest @ 1.255 V: 2.000 MHz / 1.964 MHz / 1.885 MHz
- highest / average / lowest @ 1.255 V: 7.67 mW / 7.43 mW / 7.11 mW





- averaged results from full lot of chips (130 nm)
- bars show min/max

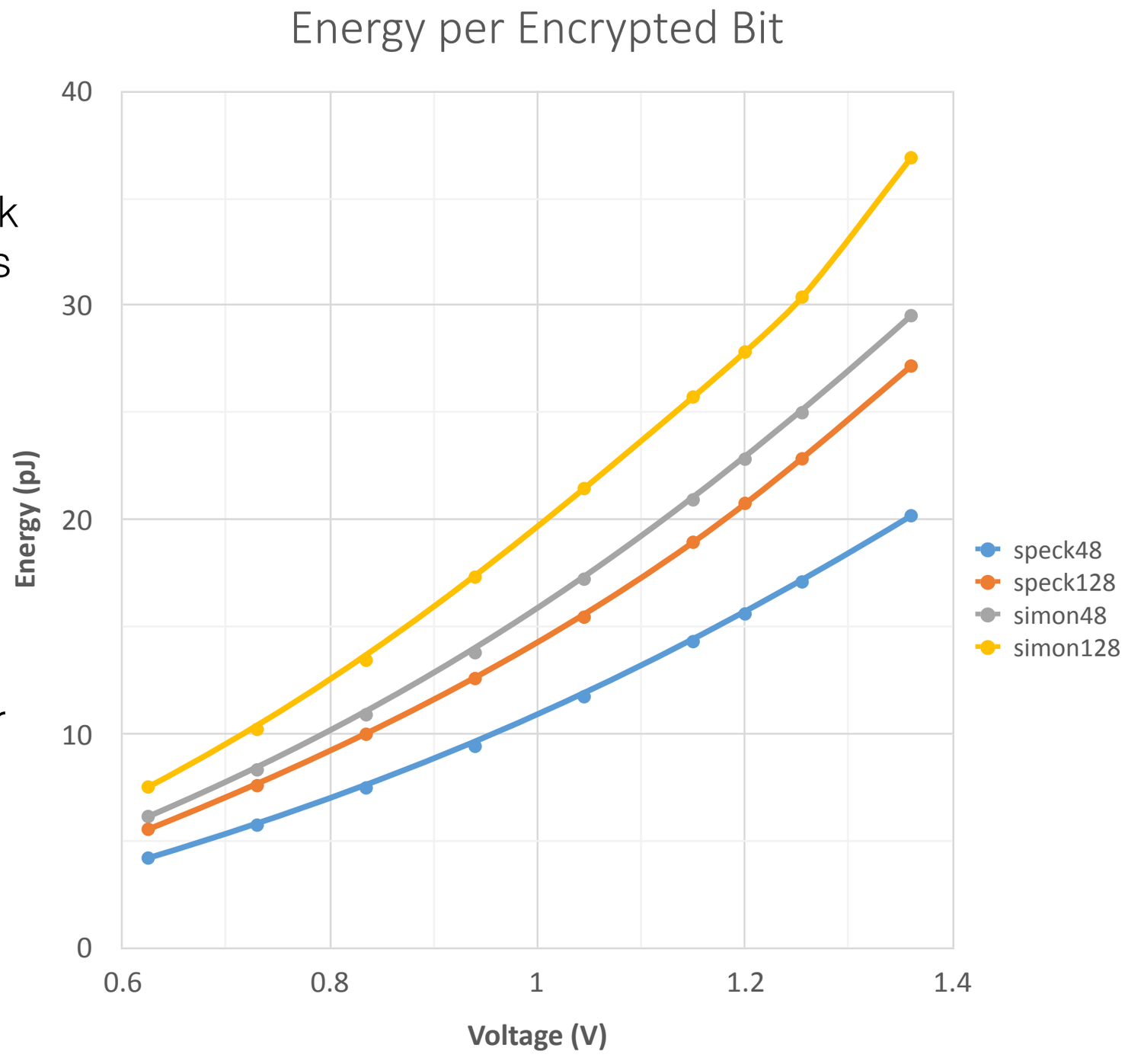




- estimated energy at 65 nm (40% reduction over two generations)

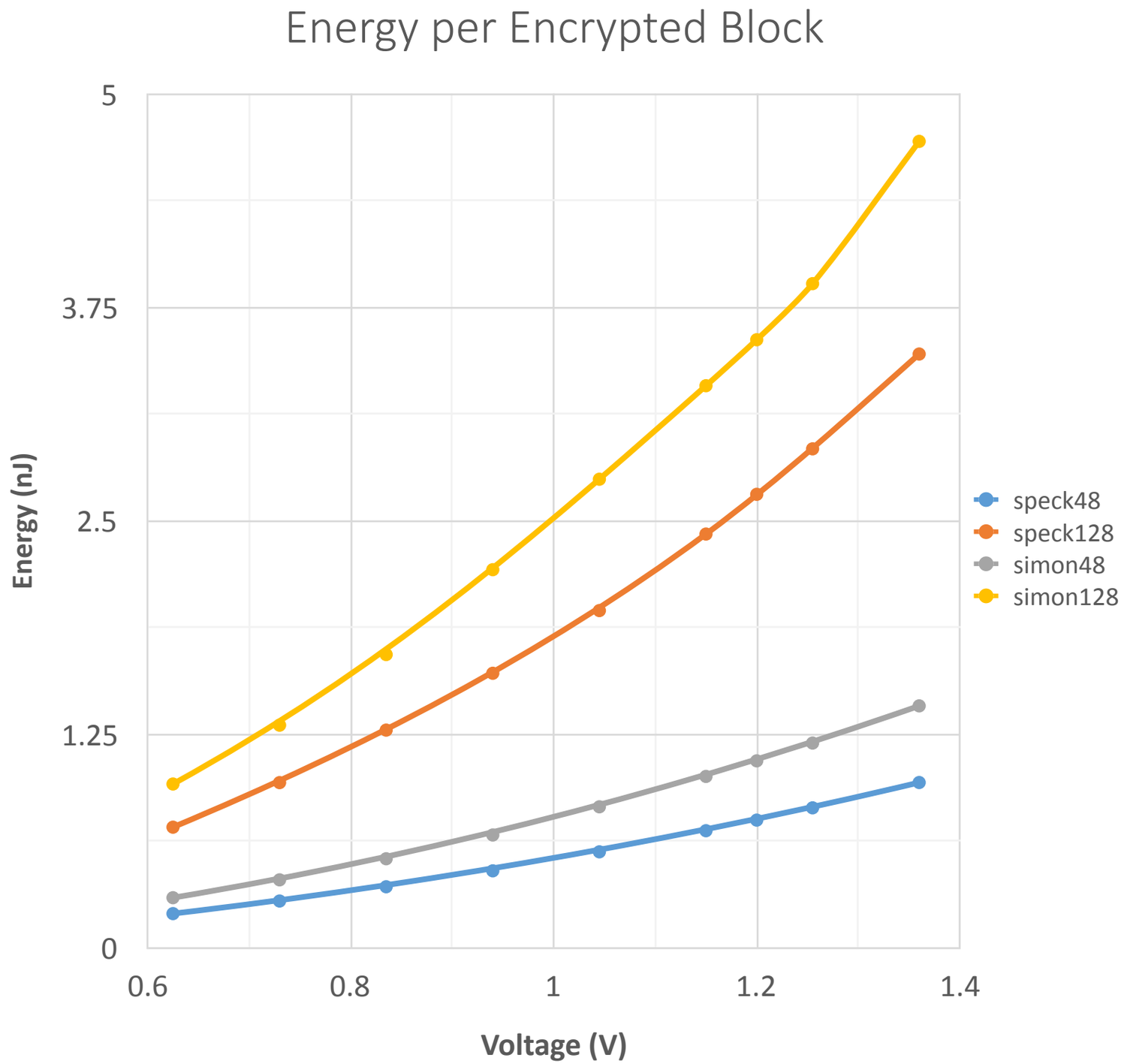
Energy Consumption

- metric: energy per encrypted bit
  - abstracts differences in key/block size as well as the internal cycles of each cipher
- original paper: "*Simon has been optimized for hardware*"
  - both Speck implementations consume less energy / bit
- ciphers are more efficient near their minimum voltage
  - ~3.6x decrease in energy consumption through voltage scaling (0.625 V vs 1.2 V)



Energy Consumption

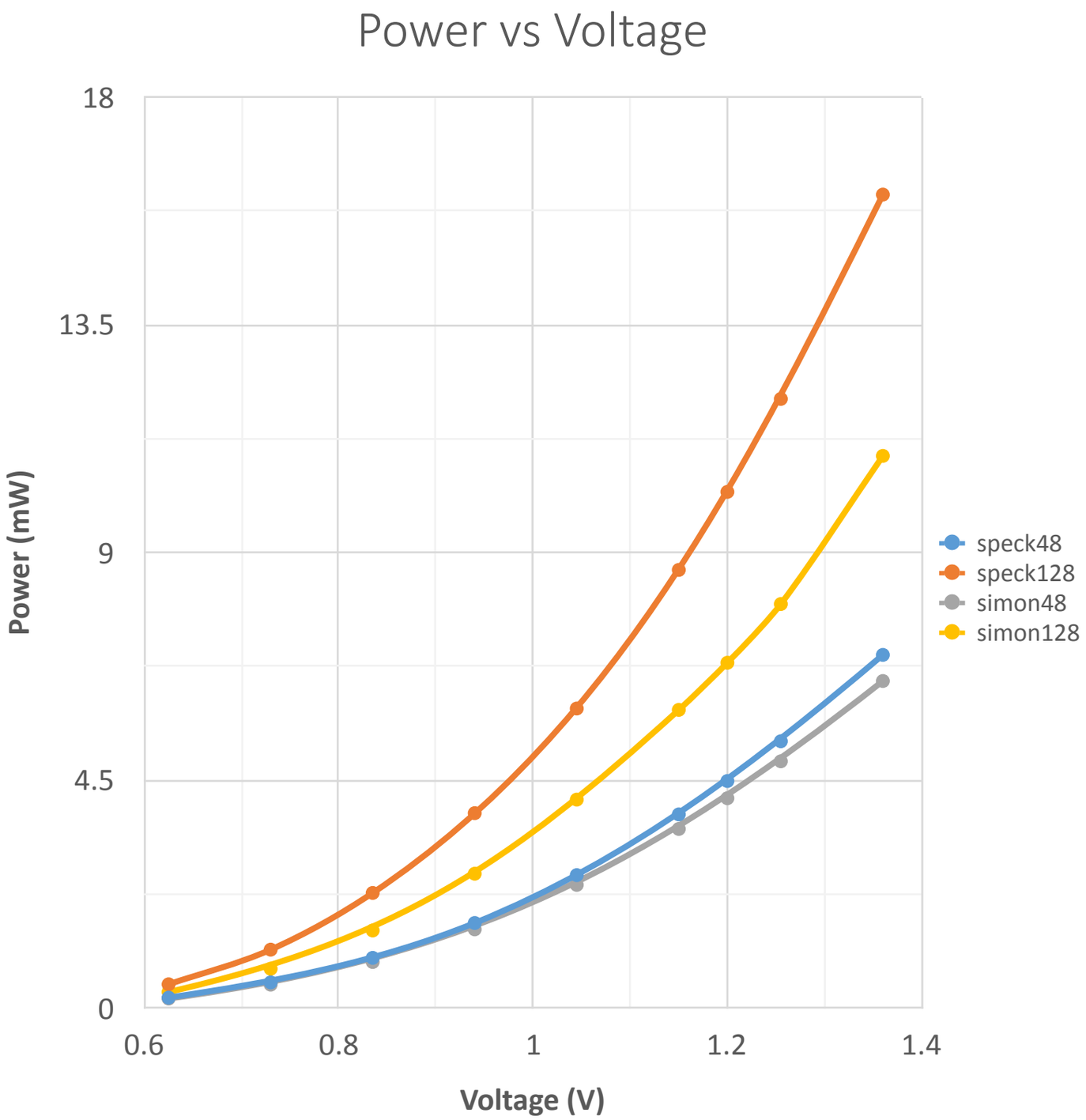
- metric: energy per encrypted **block**
- favors smaller block sizes



Power

- Speck 128/128 uses the most power during operation but is more efficient than Simon 128/128 due to fewer internal cycles

- Similarly, Speck 48/72 uses more power than Simon 48/72



Hardware Comparisons

- original paper:
  - area is shown in “gate equivalent”
  - 1 GE = 5.76 μm^2
  - throughput measured at 100kHz
- our designs:
  - much larger, but much faster

	Area (μm^2)	Area (GE)	Throughput (kbps)
simon128	62,205	10,799	250,643
simon48	41,683	7,237	192,537
speck128	111,428	19,345	523,377
speck48	43,547	7,560	299,112
aes128	1,835,174	318,607	3,678,374

GULPHAAC Designs (at 1.2 V)

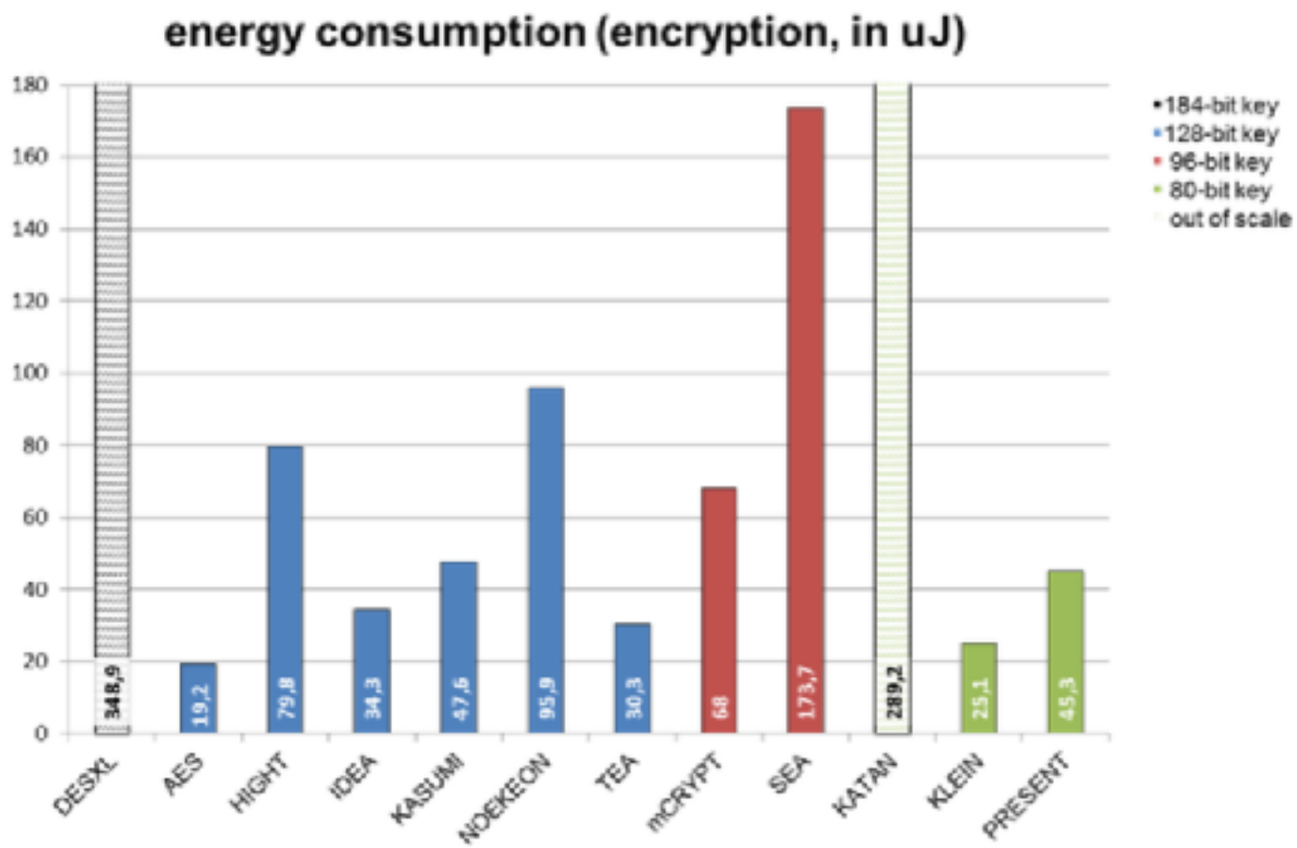
Table 6.2: Hardware performance for Simon and Speck.

algorithm	area (GE)	throughput (kbps)	algorithm	area (GE)	throughput (kbps)
Simon 48/72	631	5.1	Speck 48/72	693	4.3
	639	10.3		752	8.5
	648	15.4		777	12.8
	662	20.5		821	17.0
	683	30.8		848	25.5
	714	41.0		963	34.0
	765	61.5		1040	51.1
	918	123.1		1152	192.0
Simon 128/128	1234	2.9	Speck 128/128	1280	3.0
	1242	5.7		1338	6.1
	1263	11.4		1396	12.1
	1317	22.9		1488	24.2
	1430	45.7		1711	48.5
	1665	91.4		2179	97.0
	2090	182.9		2727	376.5

[Beaulieu et al. 2013]

Lightweight on Embedded

- unclear if energy is per block or bit
- in either case, we are significantly lower (no surprise)
- ASIC vs µC implementation



Compact Implementation and Performance Evaluation of Block Ciphers in ATtiny Devices

[Eisenbarth, 2012]

Green Crypto

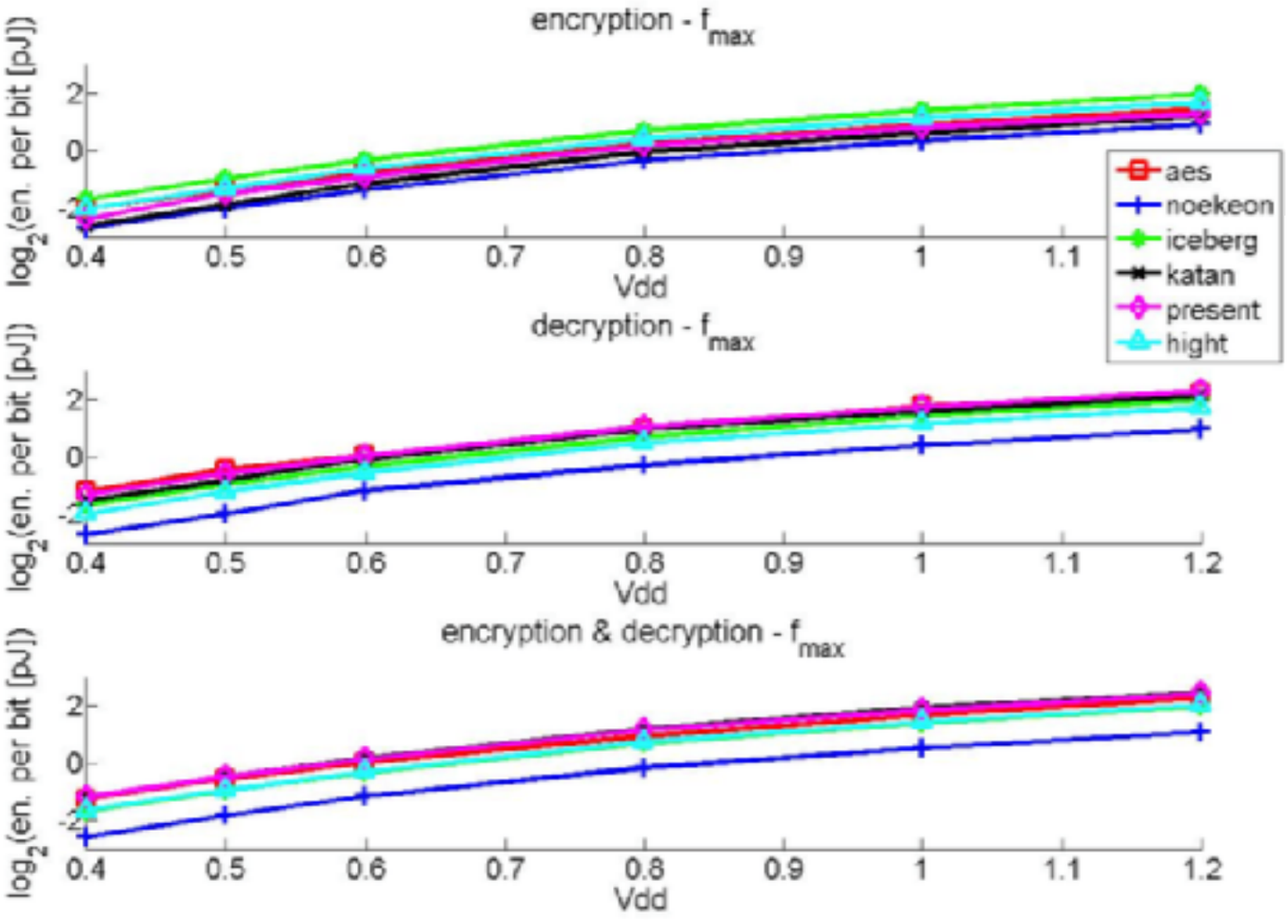


Fig. 15. Voltage scaling: energy per bit

Table 1. Implementation results for most "globally efficient" architectures

Cipher	Mode E,D,ED	Area [μm^2]	f_{max} [MHz]	Latency [cycles]	Throughput [Mbps]	Power [mW]	Energy [pJ per bit]
AES	E	17921	444	12	4740	13,5	2,9
$N_r = 1$	D	20292	377	22	2195	10,6	4,8
	ED	24272	363	≈17	≈2997	≈12,6	≈4,4
NOEKEON	E	8011	1149	18	8173	15,0	1,8
$N_r = 1$	D	10431	1075	19	7243	14,1	1,9
	ED	10483	1075	≈18,5	≈7445	≈15,35	≈2,1

Towards Green Cryptography: A Comparison of Lightweight Ciphers from the Energy Viewpoint

[Kerckhoff et al. 2012]

Open Questions

- how much synthesis pipeline tuning is necessary for dramatically different backends (e.g., BSV or SV)?

- how much intelligence to build in for optimization (e.g., automatically measuring of, and learning from, LLVM backend behavior) and parallelization and pipelining?

- what other kinds of assurance artifacts matter to customers (most folks do not understand proof, but do understand testing—how do we accommodate?)