CESEL: Flexible Crypto Accelerator for IoT

Kevin Kiningham
Dan Boneh, Mark Horowitz, Phil Levis

Background

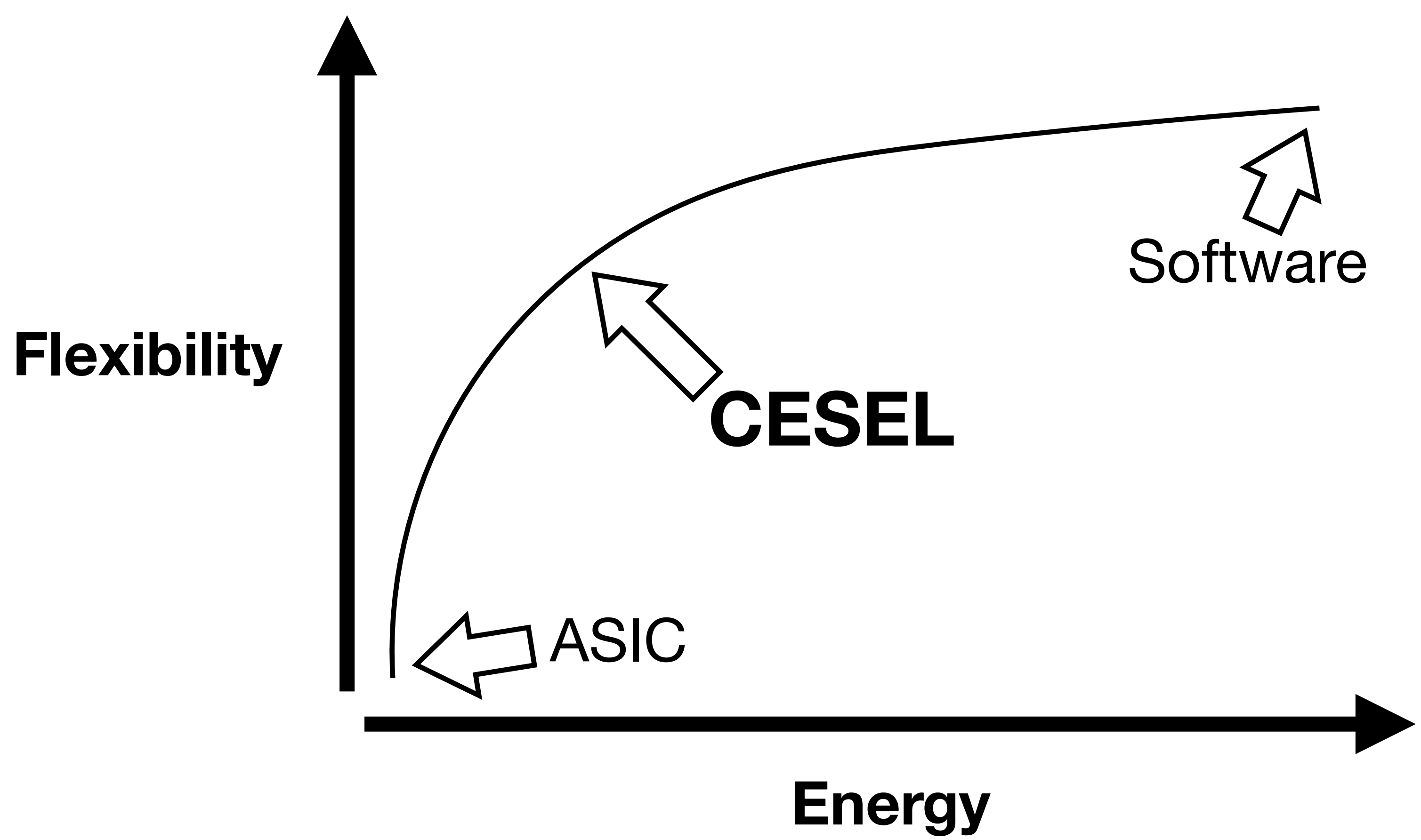
- Cryptography is critical for IoT security
  - Code signing, data privacy, user authentication, …
  - TLS, WiFi, BLE, Thread, …
- However, cryptography can use a lot of energy
- Existing Solution: Fixed function hardware accelerators
  - Implement crypto algorithm directly in silicon
  - Huge energy savings, but inflexible

The Problem

- Security requirements change over time
  - Market requirements change
  - Crypto gets broken
- Long deployments mean change is more likely
- **Current accelerators can't adapt to new crypto**
  - New cipher requires physical replacement of device

Flexible Acceleration

- We need **flexible** acceleration
  - Works for wide range of crypto
  - Gives “good enough” power reduction

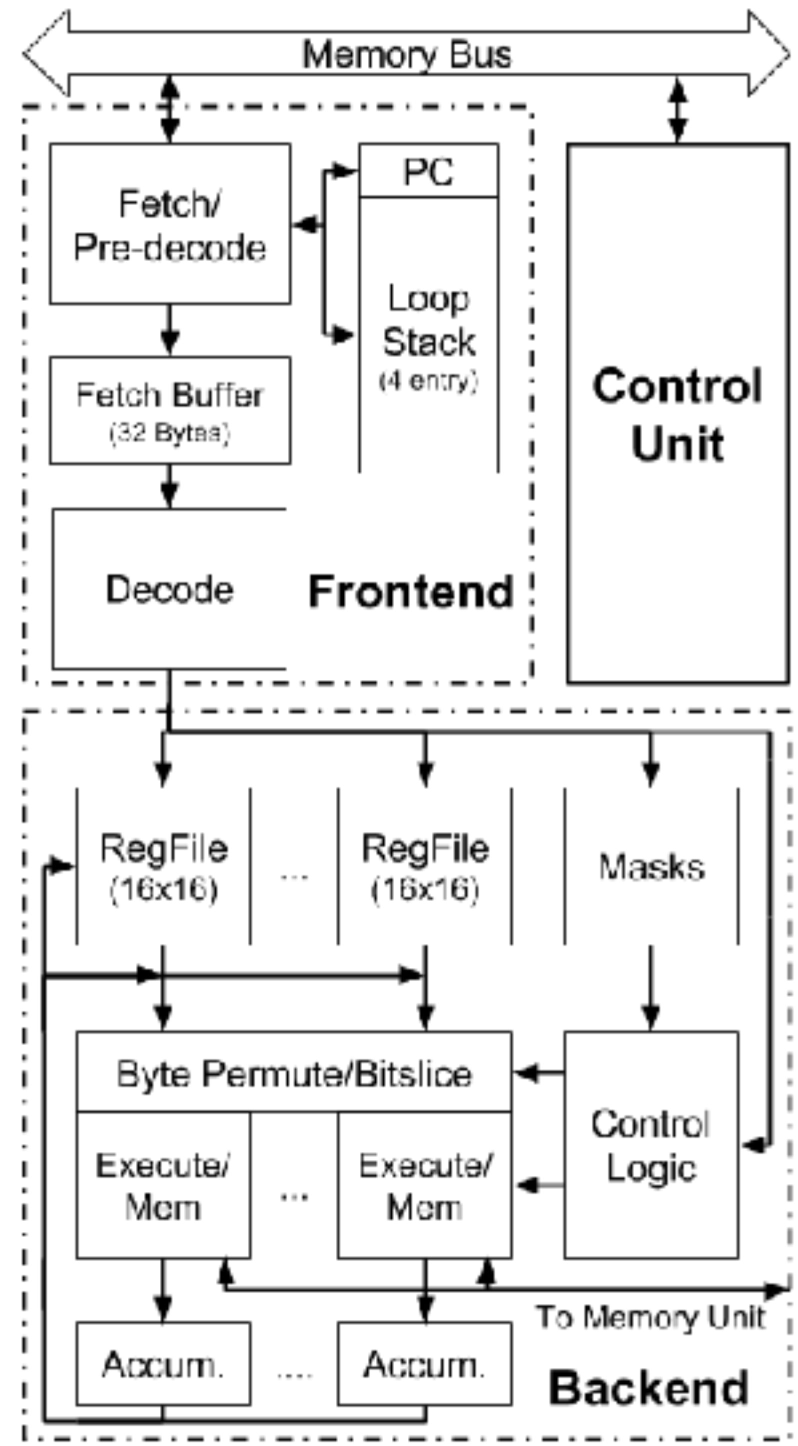


Crypto Commonalities

- Investigated 38 crypto algorithms
  - Drawn from major protocols/libraries/competitions
- Symmetric ciphers/Hash functions
  - Bit/Byte permutations
  - Internally parallel
  - Operation width <= 64 bits
- Asymmetric ciphers
  - Long-word arithmetic operations (>128 bits)

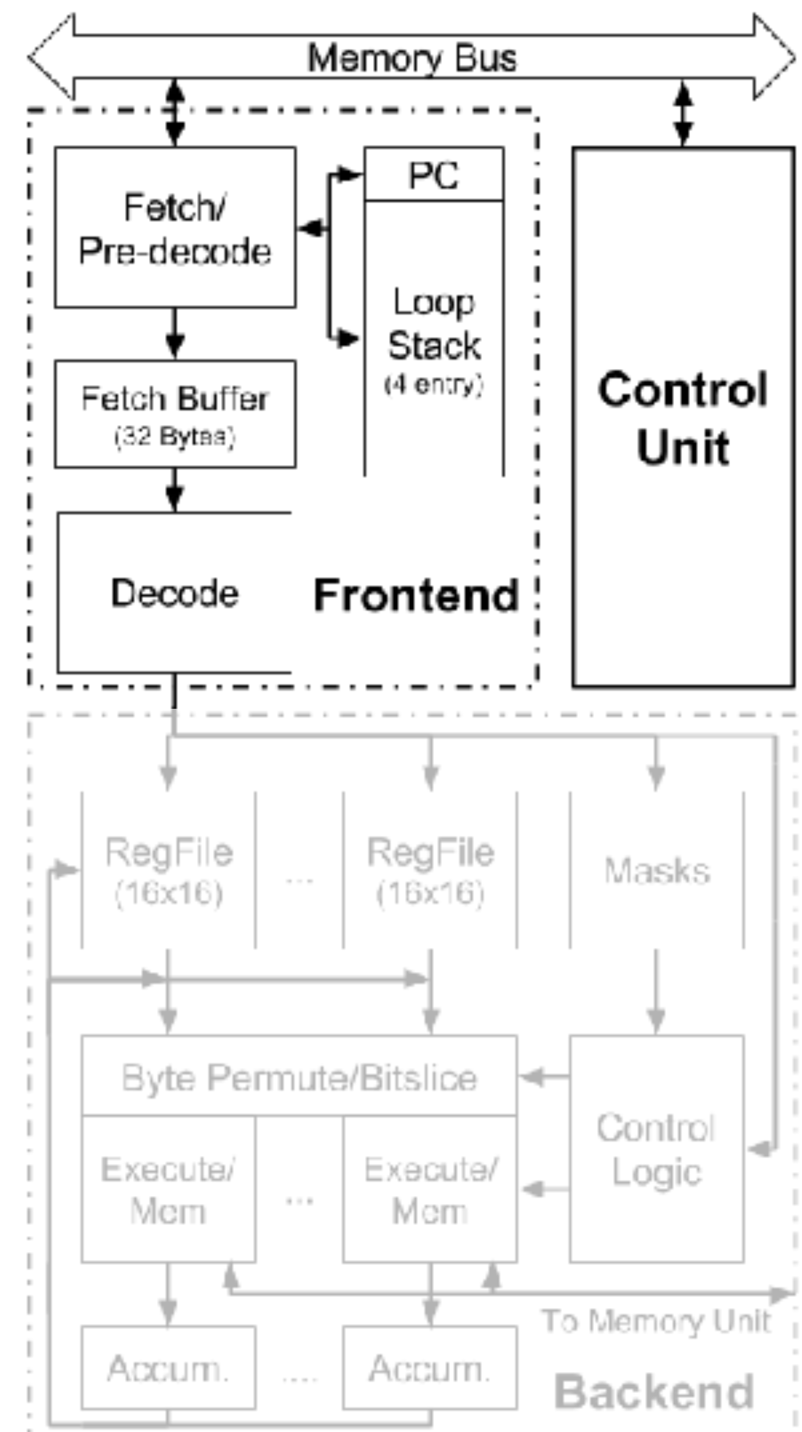
CESEL Overview

- In order SIMD architecture with 256-bit data path
- Designed to execute as co-processor
- Split into Frontend (fetch/decode) and Backend (execute/writeback)
- Number of lanes and lane width is flexible



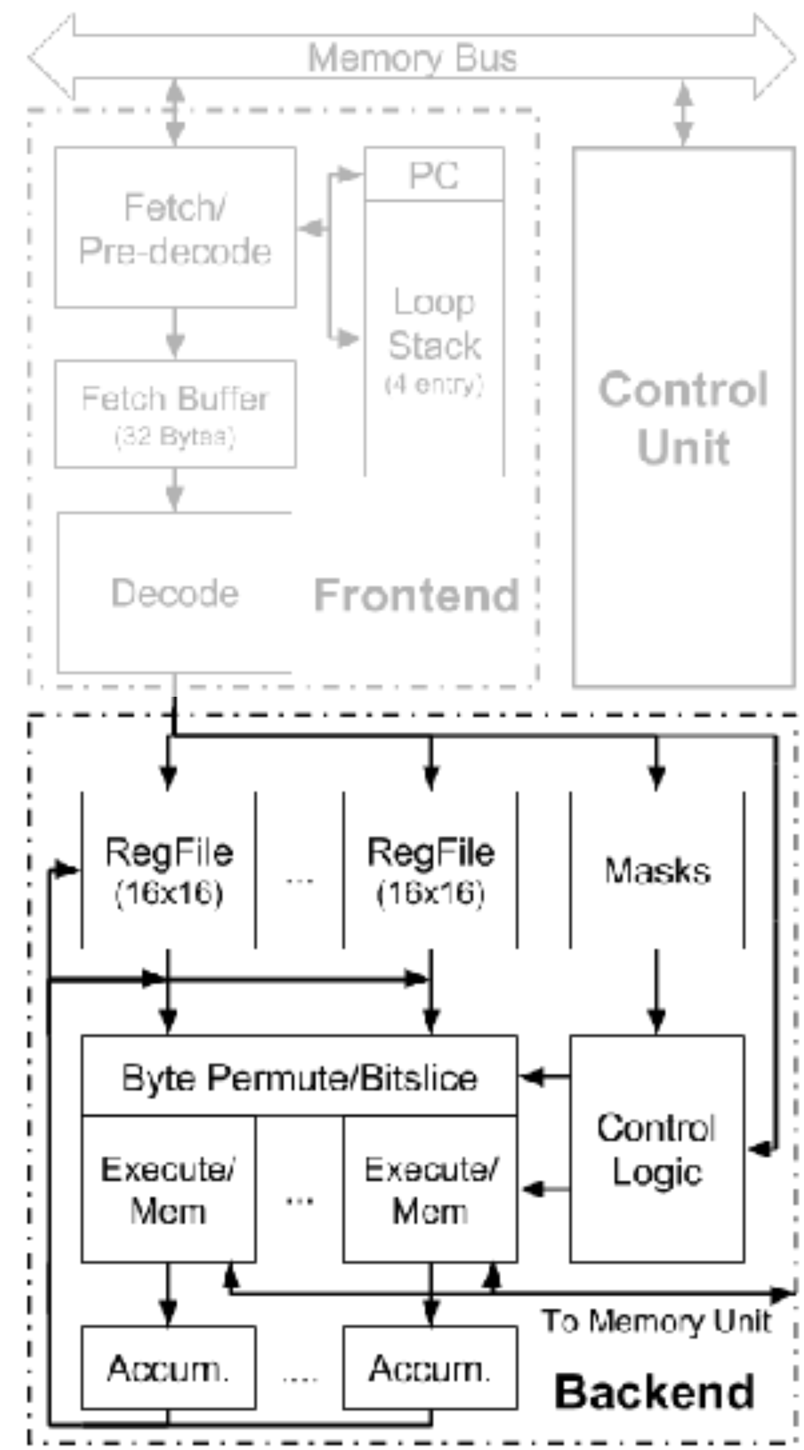
CESEL Frontend

- No data dependent control flow
  - Not needed in vast majority of crypto code
  - Simplifies implementation
- Loops use hardware “Loop Stack”
  - Hardware keeps track of loop iteration + boundary
  - Fetch stage can always predict next instruction
- 16-bit instructions
  - Minimizes energy cost of instruction fetches (significant in practice)



CESEL Backend

- SIMD lane width is flexible
  - 32x8 bits up to 1x256 bits
  - Set by special instruction during execution
- Internally, each operation mapped onto 16-bit execution units
- Fast byte permutation + bitslice
  - Useful in many cipher implementations
  - Turns bitwise operations into bytewise



Results

- Implemented CESEL in 180nm TSMC
- Measured total energy for multiple ciphers
- RISC-V CPU as baseline
  - **3.8-60x** improvement
- ~20x more energy than dedicated ASIC

	ASIC	CESEL	RISC-V
Bitsliced AES	7.07 **(0.05x)**	147.5 **(1x)**	9,036 **(60x)**
SHA2	-	3,279 **(1x)**	12,360 **(3.8x)**
ChaCha	15.3 **(0.05x)**	302.4 **(1x)**	2,021 **(6.7x)**
Curve 25519	-	8,163 **(1x)**	40,454 **(5.0x)**
RSA	-	16,840 **(1x)**	87,401 **(5.2x)**
R-LWE	-	19,822 **(1x)**	109,502 **(5.5x)**

Total estimated energy in nJ

Results

- Does this improvement matter?
- Estimated energy savings in real IoT application
  - Sensor collecting/transmitting ~1KB over BLE
  - Curve25519 key exchange once per week

	No Crypto	With CESEL	RISC-V Only
Application	3100	3100	3100
Crypto	0	460	2300
Total	3100 **(0.89x)**	3560 **(1x)**	5400 **(1.51x)**

Total estimated energy in nJ

Recap

- IoT needs crypto acceleration + flexibility
- Our solution: CESEL
  - Wide SIMD + long word support
  - Special instructions (permute, bitslice)
  - No data-dependent control flow
- Significant energy savings compared to software
  - **~5x** for most ciphers
  - **1.5x** longer deployment time

CESEL

- IoT needs crypto acceleration + flexibility

- Our solution: CESEL
  - Wide SIMD + long word support
  - Special instructions (permute, bitslice)
  - No data-dependent control flow

- Significant energy savings compared to software
  - **~5x** for most ciphers
  - **1.5x** longer deployment time