Instruction Set Extensions for Efficient AES Implementation on 32-bit Processors

Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2006

Stefan Tillich **and Johann Großschädl**

IAIK – Graz University of Technology
Stefan.Tillich@iaik.tugraz.at
www.iaik.tugraz.at

Outline

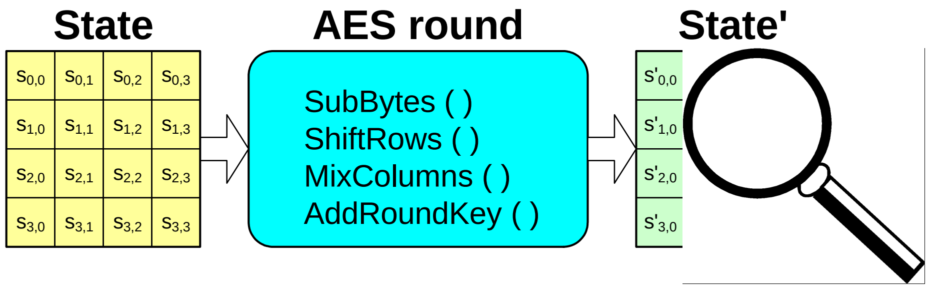
- Motivation
- Design principles
- Proposed AES instruction set extensions
- Practical results
  - Hardware cost
  - Performance
  - Code size
- Note on side-channel attacks
- Conclusions

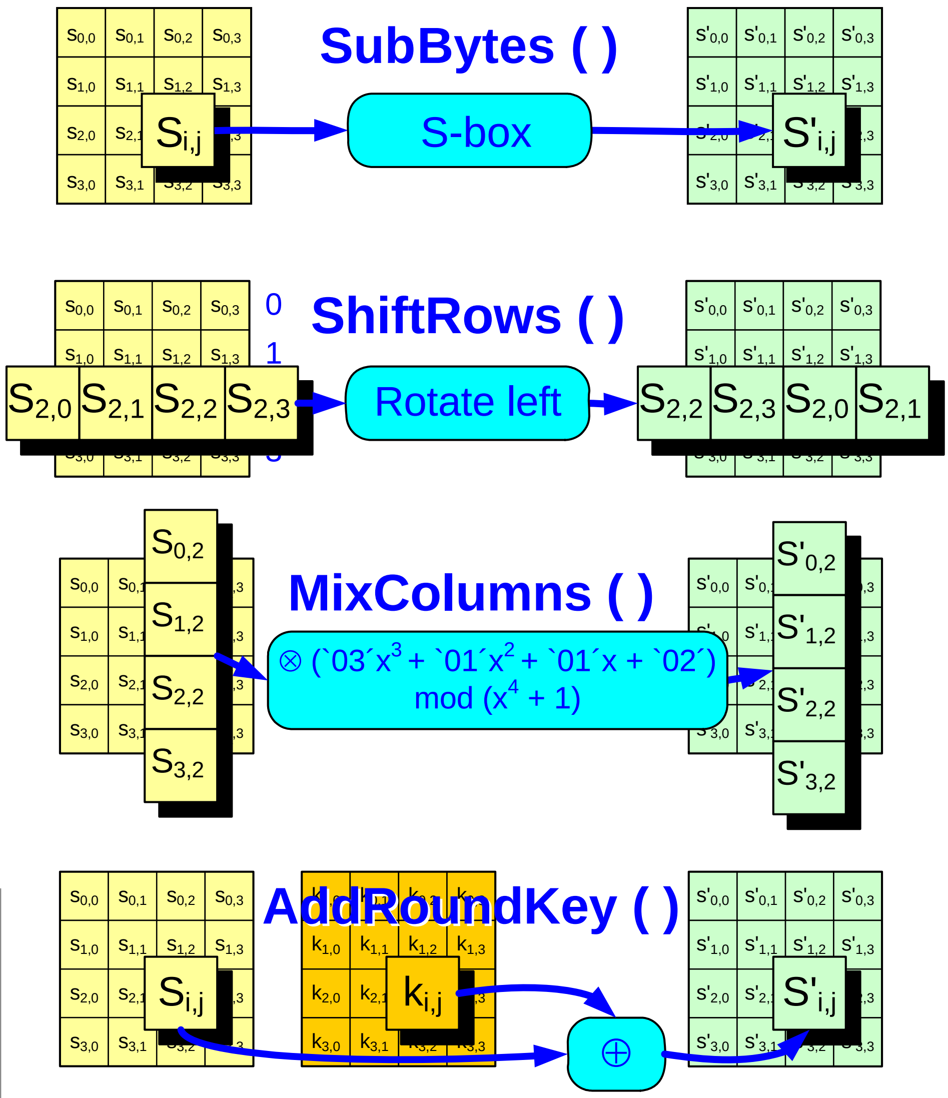
Motivation

- Mid-range to high-end embedded systems incorporate 32-bit processors
- Efficient implementation of cryptographic algorithms under different aspects
  - Performance
  - Code size, memory footprint
  - Limited energy budget
  - Flexibility, extensibility
- ISE design approach
  - Custom instructions added to general-purpose processor
  - Unites features of pure-software and dedicated hardware solutions
- Extensive study for symmetric cryptography by means of AES

AES Algorithm

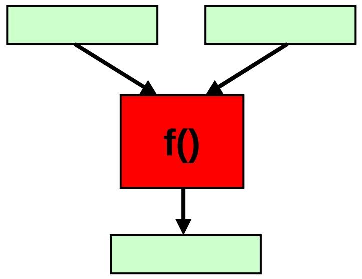
- 128, 192, or 256-bit key
- 10, 12, or 14 rounds
- 4 x 4 byte State
- 4 transformations
  - SubBytes
  - ShiftRows
  - MixColumns
  - AddRoundKey




SubBytes ( )
S-box
$S_{i,j}$
$S'_{i,j}$
ShiftRows ( )
Rotate left
0
1
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,1}$
MixColumns ( )
$\otimes$ (`03´$x^3$ + `01´x^2 + `01´x + `02´) mod (x^4 + 1)
$S_{0,2}$ $S_{1,2}$ $S_{2,2}$ $S_{3,2}$
$S'_{0,2}$ $S'_{1,2}$ $S'_{2,2}$ $S'_{3,2}$
AddRoundKey ( )
$k_{i,j}$
$\oplus$

Design Principles for AES Extensions

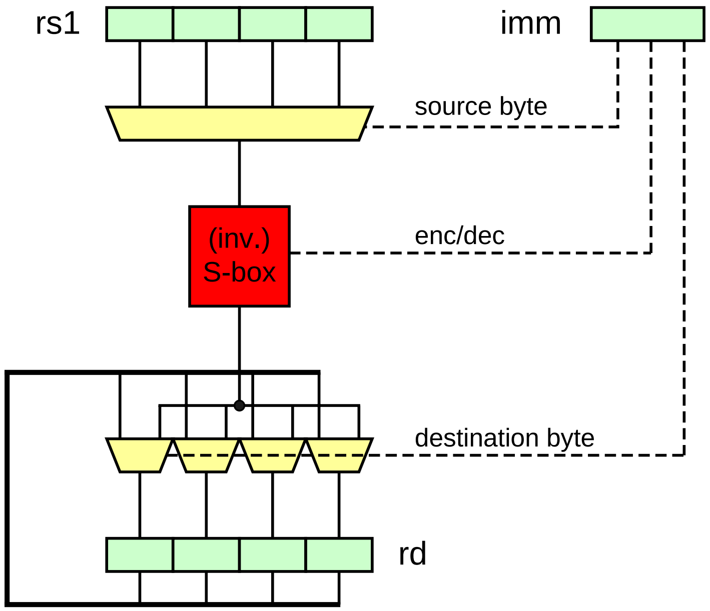
- Custom instructions to support round transformations and key expansion
- For all key sizes, modes of operation, etc.
- Easy integration in 32-bit embedded processors
- RISC-like instruction format (2 input operands, 1 output operand)
- No non-standard architectural features (e.g. dedicated lookup tables)
- Short critical path

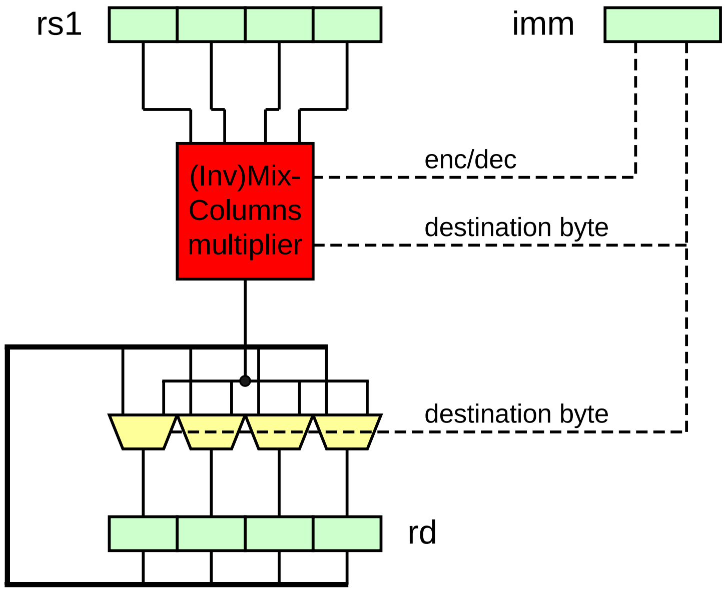


Byte-Oriented AES Extensions

- Low-cost instructions
- 1st operand is register,
  2nd operand is immediate value
- Immediate value determines operation
- Produce 1-byte result
- Result is written to a specified byte of destination register

Byte-Oriented AES Extensions





- **`sbox rs1, imm, rd`**
- Supports SubBytes, ShiftRows, key expansion

- **`mixcol rs1, imm, rd`**
- Supports MixColumns

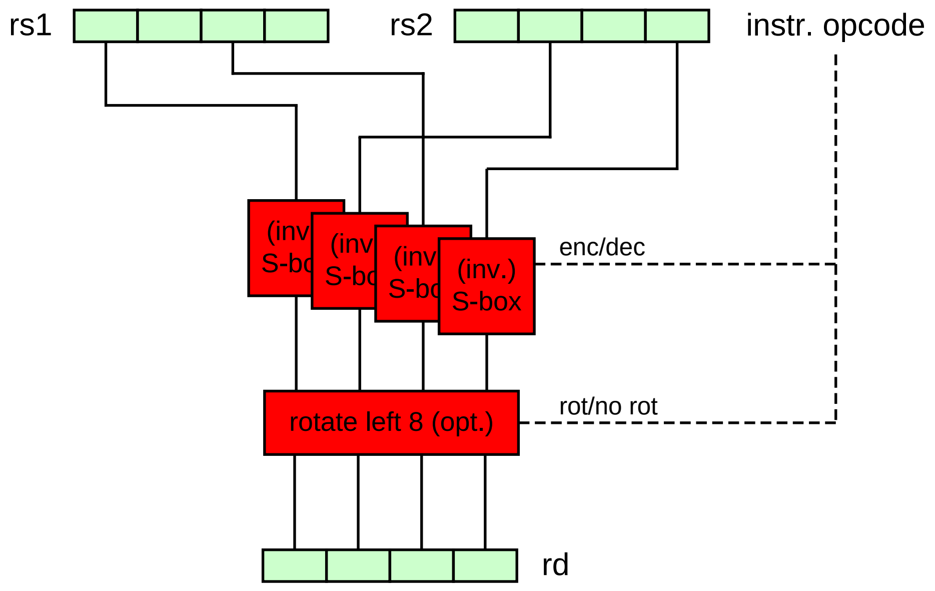
Plain Word-Oriented AES Extensions

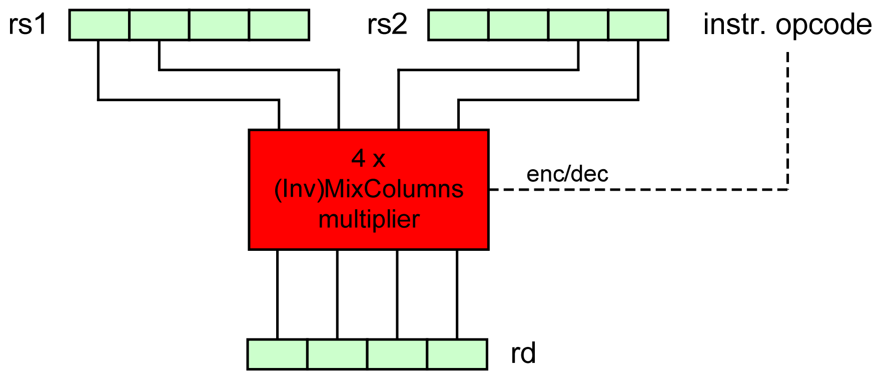
- How can performance be improved?
- Simple idea: Functionality of byte-oriented extensions quadrupled
- **`sbox`** -> **`sbox4`**, **`mixcol`** -> **`mixcol4`**
- Problem: **`sbox4`** & **`mixcol4`** cannot be combined efficiently
  - ShiftRows requires rows, MixColumns requires columns
  - ShiftRows becomes bottleneck!

Solving the "ShiftRows Problem"

- Pack AES State columns into 32-bit registers
- Work on 2 State columns simultaneously
- Split ShiftRows into 2 parts; performed implicitly
  - 1st part at end of SubBytes
  - 2nd part at beginning of MixColumns
- Implementation
  - Instructions have two register source operands (2 State columns)
  - Appropriate bytes selected for transformation

Advanced Word-Oriented AES Extensions





- **`sbox4s  rs1, rs2, rd`**
- **`isbox4s rs1, rs2, rd`**
- **`sbox4r  rs1, rs2, rd`**
- Supports SubBytes, ShiftRows, key expansion

- **`mixcol4s  rs1, rs2, rd`**
- **`imixcol4s rs1, rs2, rd`**
- Supports ShiftRows, MixColumns

Implementation

- AES extensions integrated into SPARC V8-compatible Leon2 embedded processor
- Estimation of hardware overhead
  - Synthesis of complete 5-stage pipeline (integer unit) using UMC 0.13 µm standard-cell library
  - 4 ns critical path delay (i.e. 250 MHz)
- Performance evaluation
  - Prototyped modified processor on FPGA board
- Code size
  - Compilation with GNU toolchain

Hardware Cost

Integer unit variant	***Gate equivalents***	***Area increase***
Baseline (no extensions)	13,349	-
Low cost (**`sbox`** & **`mixcol`**)	13,853	+ 4%
Area/perf. trade-off (**`sbox`** & **`mixcol4`**)	14,583	+ 9%
High performance (**`sbox4s`** & **`mixcol4s`**)	16,370	+ 23%

AES-128 Encryption (Precomputed Key Schedule) Performance and Code Size

AES impl.	***Key exp.***	***Encryption***	***Code size***
Baseline (no extensions)	1.0 (739 cycles)	1.0 (1,637 cycles)	1.0 (2,168 byte)
Low cost (**`sbox`** & **`mixcol`**)	2.1	2.8	- 72%
Area/perf. trade-off (**`sbox`** & **`mixcol4`**)	2.1	4.8	- 78%
High performance (**`sbox4s`** & **`mixcol4s`**)	2.3	8.3	- 59%
T lookup (Gladman) 4 KB table	1.7	1.5	+ 403%

AES-128 Encryption (On-the-fly Key Expansion) Performance and Code Size

AES impl.	***Encryption***	***Code size***
Baseline (no extensions)	1.0 (2,239 cycles)	1.0 (1,636 byte)
Low cost (**`sbox`** & **`mixcol`**)	4.4	- 76%
Area/perf. trade-off (**`sbox`** & **`mixcol4`**)	5.6	- 79%
High performance (**`sbox4s`** & **`mixcol4s`**)	9.9	- 48%
T lookup 4 KB table	1.5	+ 231%

Note on Side-Channel Attacks

- Not main focus of this work
- Instructions for S-box remove data-dependent table lookups
  - No cache-timing attacks possible
- Increased parallelism
  - Can increase power analysis resistance
- Masking countermeasures supported partly
  - MixColumns, masked S-box (re)computation
- Other countermeasures remain applicable
  - e.g. randomization, secure logic styles

Conclusions

- Instruction set extensions for AES for 32-bit processors
  - Highly flexbile
  - Easily implementable
  - Short critical path
- Different area/performance trade-offs possible
  - Minimal to moderate increase in area (+ 4% to + 23% of IU)
  - Speedup to a factor of nearly 10 achievable
  - Significant reduction of code size (- 84% possible)
- Interesting design option towards "zero-overhead" symmetric cryptography on embedded systems