# Hardware/Software Co-Design of Elliptic Curve Cryptography on an 8051 Microcontroller

Manuel Koschuch, Joachim Lechner, Andreas Weitzer, Johann Großschädl, Alexander Szekely, Stefan Tillich, and Johannes Wolkerstorfer

IAIK – Graz University of Technology
http://www.iaik.tugraz.at

# Motivation

- Problem
  - ECC is very computational-intensive
  - 8-bit micro-controllers like 8051 are too slow for ECC
- Goal
  - ECDSA on 8051 in less than 1 sec (scalar mult. in ≤ 0.5 sec)
  - Low-cost hardware accelerator (minimal area)
- Hardware/software co-design
  - Field arithmetic in HW, rest in SW
  - 192-bit Arithmetic Unit (approx. 11k gates)
  - Addition and Multiplication in GF($2^m$)
  - Scalar multiplication over GF($2^{191}$) in 120 msec @ 12 MHz

# Outline

- ECC
- Design approaches
- System components
  - 8051
  - Elliptic Curve Acceleration Unit (ECAU)
  - DMA
- Results
- Conclusions

# Elliptic Curve Cryptography

- Operates in a group of points on a curve defined over finite field
- Less bits than in RSA systems provide same security level
  - 160 bit (ECC) vs. 1024 bit (RSA)
- Group operation: point addition
- Scalar multiplication ($k$*P) is basic building block of ECC cryptosystems
  - Done by repeated point addition/doubling
  - Arithmetic operations in underlying finite field

# Algorithmic Implementation Options

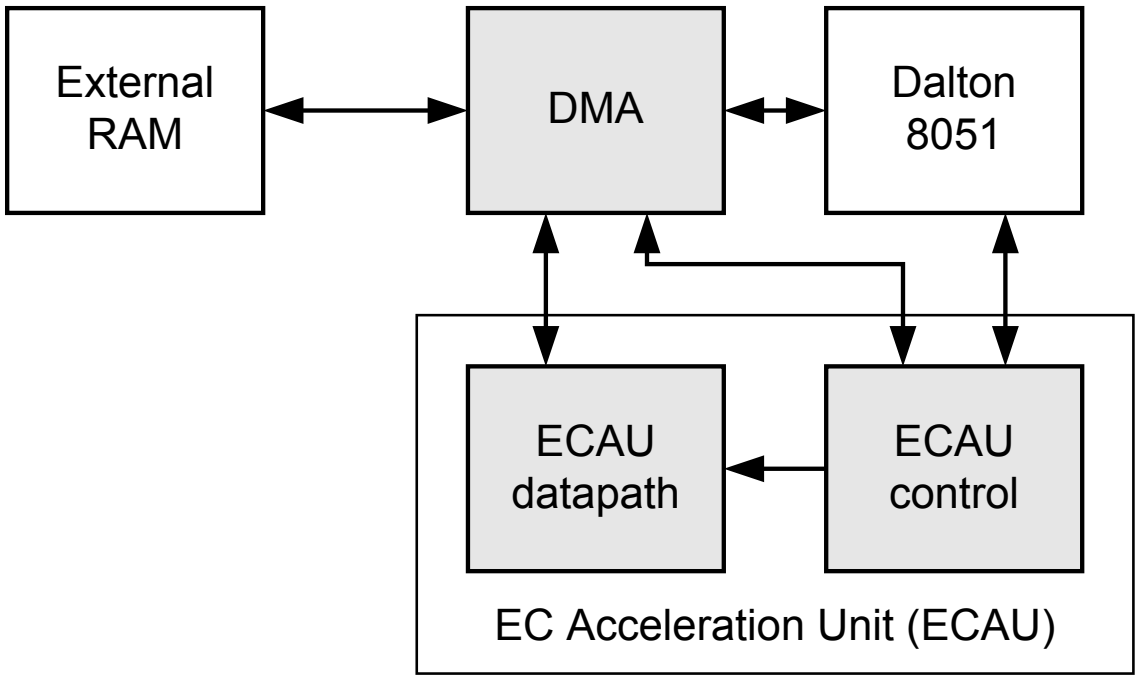
- Field type:
  - GF($p$)
  - GF($2^m$)
  - GF($p^m$)
- Representation of field elements
  - Polynomial basis
  - Normal basis
- Representation of points
  - affine coordinates
  - projective coordinates
- Algorithm for scalar multiplication

# Architectural Design Space

- Where to draw the line between HW and SW?
  - HW/SW boundary determines performance and area
- High-level exploration of design space
  - Performance evaluation with SystemC models

| Design | Hardware part | Software part |
|---|---|---|
| Variant 1 (high performance) | Group operations (point adding/doubling) | Control flow |
| Variant 2 (compromise) | Finite field arithmetic (adding, multiplying, …) | Group operations |
| Variant 3 (small silicon area) | Instruction set extensions (e.g. for field multiplication) | Field arithmetic and group operations |

# System Overview

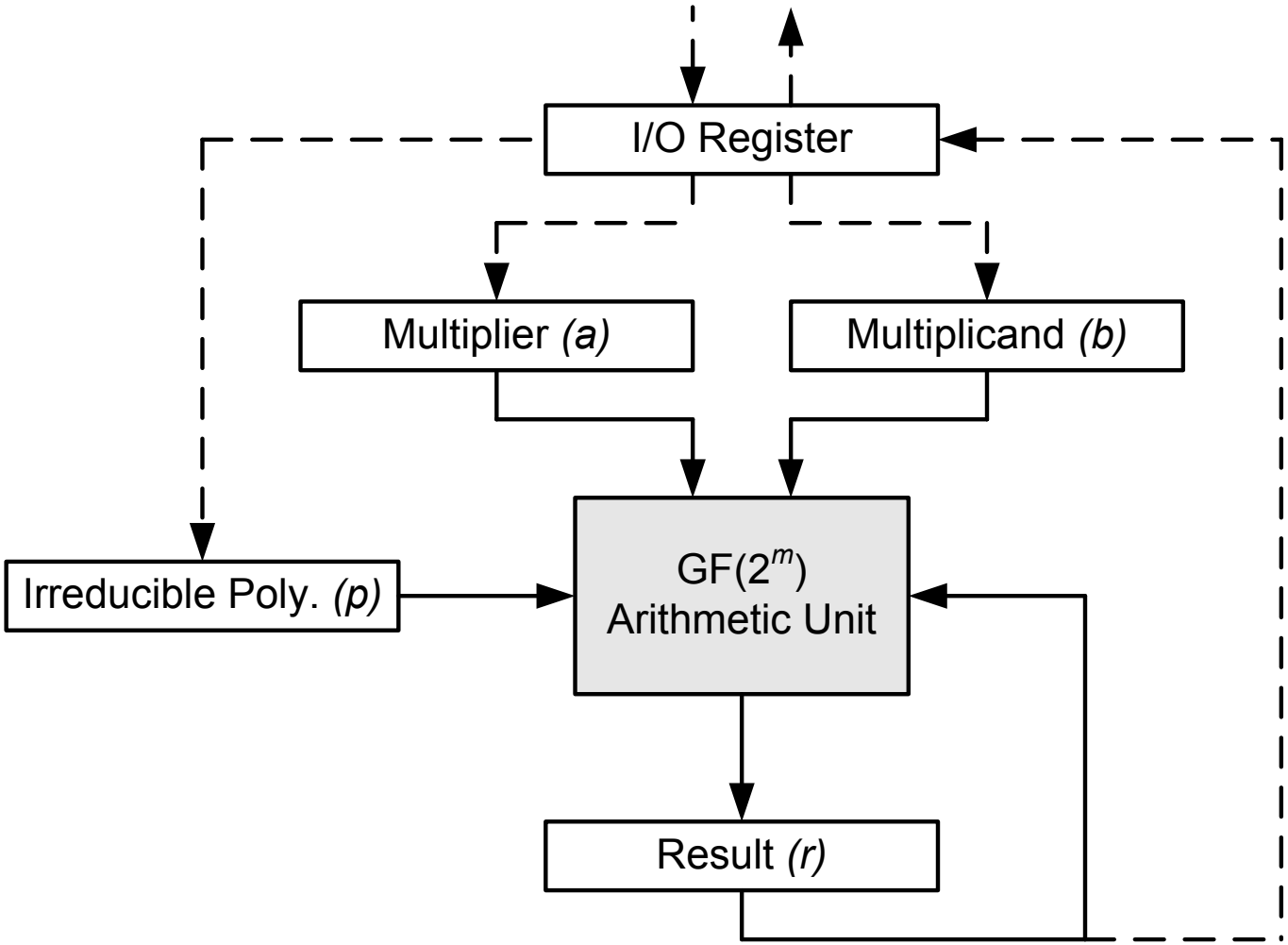



- ECAU performs field addition and field multiplication in HW
- Intermediate results during scalar mul. are stored in XRAM
- DMA for fast data transfer between ECAU and XRAM

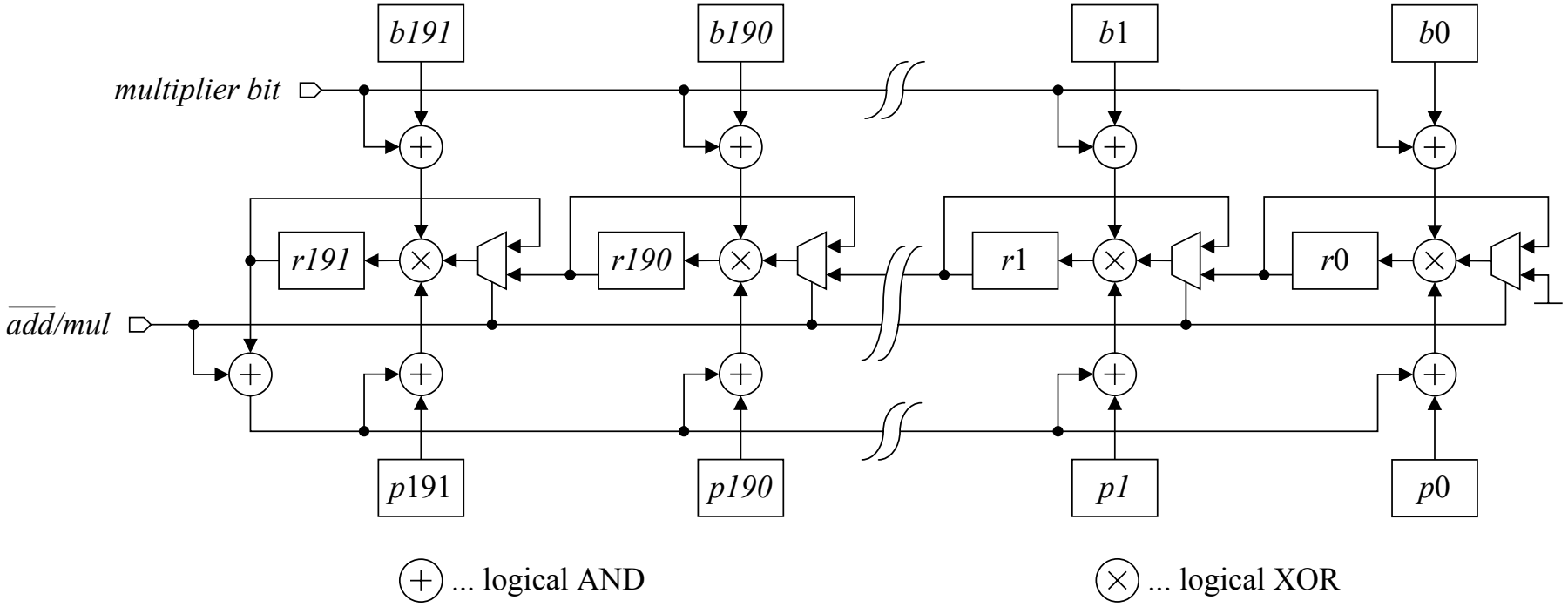
# Dalton 8051

- Developed at UC Riverside
  - GNU GPL
- 8-bit micro controller
  - 4 kbytes ROM
  - 128 bytes internal RAM + 128 bytes SFRs
  - Interface provided for 64 kbytes of external RAM
  - 12 MHz
- Limitations of the VHDL Model
  - No Interrupts
  - No peripheral devices are implemented
  - Up to 17 cycles per instruction

# ECAU


I/O Register
Multiplier (a)
Multiplicand (b)
Irreducible Poly. (p)
GF(2^m)
Arithmetic Unit
Result (r)

# Bit-Serial MSB-First Multiplier




- Multiplier works for operands ≤ 192 bit, e.g. GF($2^{191}$), GF($2^{163}$)
- All operands are left-aligned in the registers
- A multiplication in GF($2^m$) takes $m$ cycles, addition 1 cycle

# Data Transfer

- Intermediate results stored in XRAM
- Problem: how to transfer operands from Dalton to ECAU?
  - Determines overall performance
- Original Dalton provides
  - I/O Ports
  - Serial Interface
- Bottleneck memory access
  - > 30 cycles needed to transfer a single byte
  - > 700 cycles for transfer of a 191-bit operand

# DMA

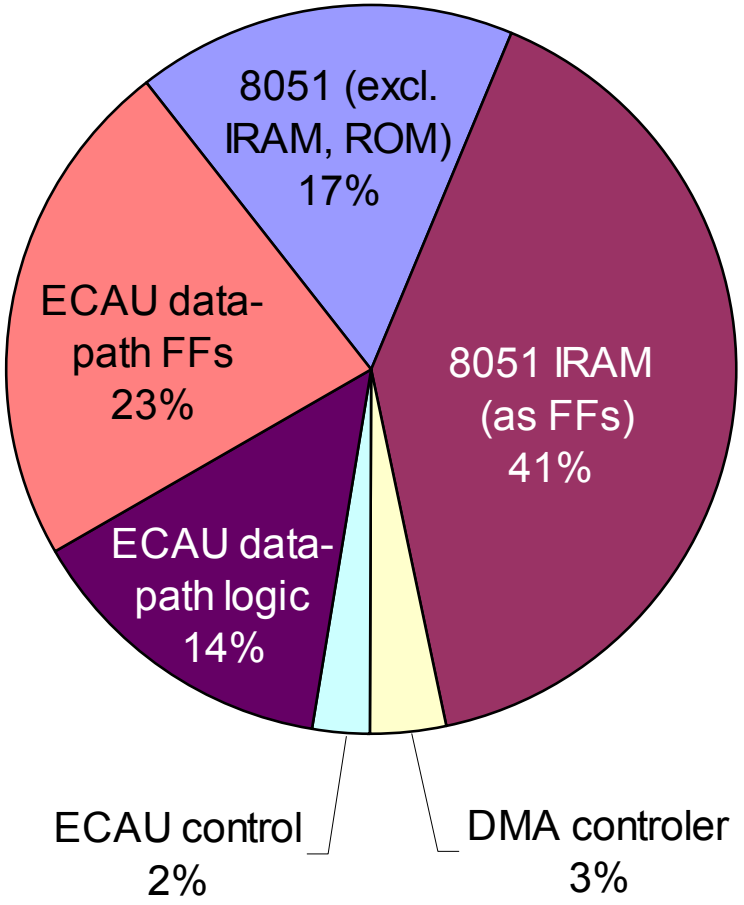
- Integration of DMA
  - Fast Operand transfer between ECAU and XRAM
- Complete 191-bit operand transfer now completed in 85 cycles
- Allows for certain amount of flexibility by left-aligning operands (bytewise)
  - 163-bit operand transfer faster than 191-bit
- In normal operation mode completely transparent for Dalton

# Results (Operations)

| Operation | Cycles |
|---|---|
| Transfer of one 191-bit operand | 85 |
| Addition in GF($2^{191}$) excl. operand transfers | 4 |
| Multiplication in GF($2^{191}$) excl. operand transfers | 195 |
| Point addition incl. operand transfers | 2.623 |
| Point doubling incl. operand transfers | 2.623 |
| Full scalar multiplication over GF($2^{163}$) | 1.190.000 |
| Full scalar multiplication over GF($2^{191}$) | 1.416.000 |

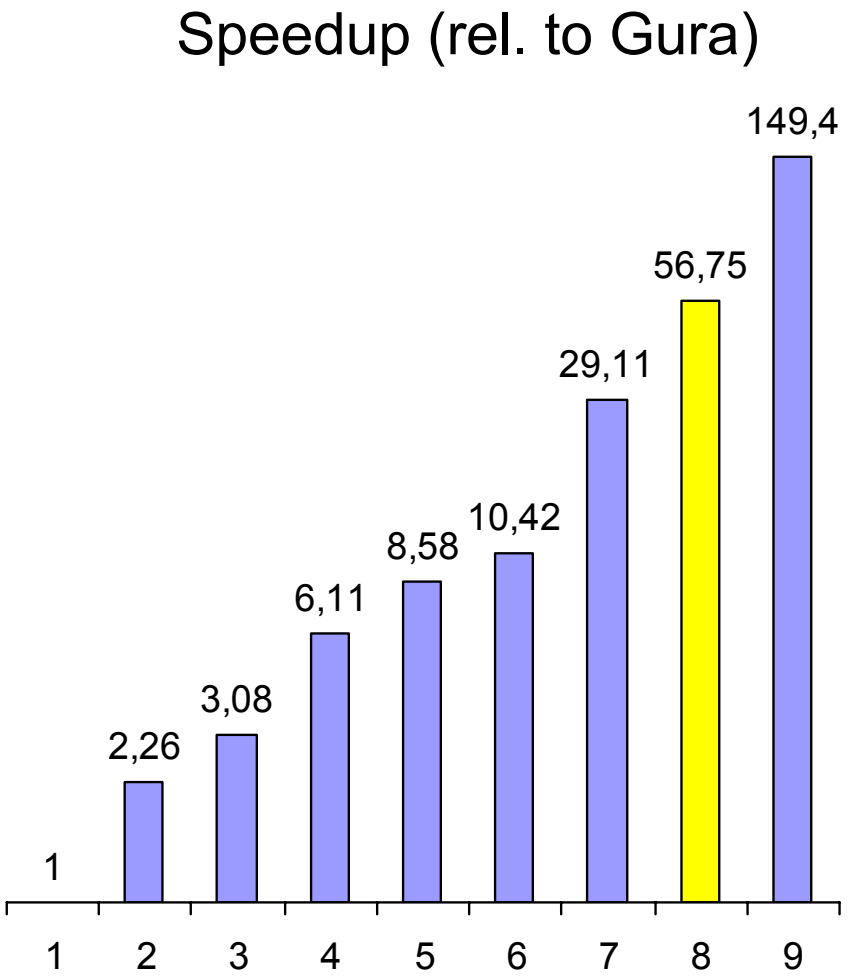
# Results (Hardware Cost)

| Component | Size (GE) |
|---|---|
| 8051 (excl. IRAM, ROM) | 4.984 |
| IRAM (as flipflops) | 11.860 |
| DMA | 1.029 |
| ECAU control | 718 |
| ECAU datapath logic | 4.180 |
| ECAU datapath flipflops | 6.720 |



# Results (Performance)

| Reference | System | Security | Cycles |
|---|---|---|---|
| Gura | 8051 | 160 bit | 67,530M |
| Batina | 8051 | 166 bit | 29,860M |
| Woodbury | 8051 | 134 bit | 21,960M |
| Kumar | 8051 | 134 bit | 11,060M |
| Hodjat | 8051 | 166 bit | 7,870M |
| Gura | ATmega128 | 160 bit | 6,480M |
| Eberle | ATmega128 | 163 bit | 2,320M |
| This work | 8051 | 163 bit | 1,190M |
| Kumar | AT94K | 163 bit | 0,452M |



# Comparison with Batina et al (CHES 05)

| Component | Size (norm.) | Area-delay product (Size * msec) | Code size (Bytes) | XRAM (Bytes) |
|---|---|---|---|---|
| Dalton 8051 | 1,00 | | | |
| Batina (166 bit HECC) | 1,15 | 2.861,20 | 11.524 | 936 |
| This work (163 bit) | 1,75 | 173,60 | 2.568 | 336 |
| This work (191 bit) | 1,75 | 206,50 | 2.568 | 384 |

# Conclusions

- ECC can benefit from significant speedup when using dedicated hardware
- Minimalist HW accelerator for field arithmetic allows 50x performance gain over SW
- System-level bottlenecks like operand transfers must be considered too
- No use to speed up only ECC when operand transfer is too slow
- HW/SW co-design to optimize for both performance and area