



# **Design of Instruction Set Extensions and Functional Units for Energy-Efficient Public-Key Cryptography**

*Johann Großschädl and Stefan Tillich*

ECRYPT Workshop on RFID & Lightweight Crypto  
Graz, July 14-15, 2005

- ◆ **Lightweight cryptography**
  - RFID tags, sensor nodes, smart cards ...
  - **Low power/energy** is premier design goal
  - What can a hardware designer do for low power?
- ◆ **Low-power crypto hardware**
  - Example: **unified multiplier** for elliptic curve crypto
  - Same datapath for **integers and binary polynomials**
  - Many **implementation options**: array/tree, radix 2/4
  - How do these options impact the energy?
  - Energy-efficiency of integers vs. binary polynomials?

- ◆ **Elliptic curve cryptography**
- ◆ **Multiplication in  $GF(p)$  and  $GF(2^m)$**
- ◆ **Unified multiplier**
  - Same datapath for integers & binary polynomials
  - Generation of partial products (radix-2 vs. radix-4)
  - Addition of partial products (dual-field adder)
  - Final adder
- ◆ **Prototype implementation**
- ◆ **Simulation results (area, delay, power)**

## ◆ Elliptic curve over a field $K$

- Weierstraß equation:  $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$
- EC = Set of all points  $(x, y) \in K \times K$  satisfying WE
- **Group structure** can be embedded
- Group operation: **addition of points**
- **Point multiplication**:  $k \cdot P = P + P + P + \dots + P$
- Realized by arithmetic operations in  $K$

## ◆ Finite field $\mathbf{GF}(q)$ , $q = p^m$

- Finite set of elements with 2 operations:  $+$ ,  $\bullet$
- **Prime fields**  $\mathbf{GF}(p)$ , **binary extension fields**  $\mathbf{GF}(2^m)$

## ◆ Prime field $\text{GF}(p)$

- Addition and multiplication modulo the **prime**  $p$
- E.g.:  $\text{GF}(7) = \{0, 1, 2, 3, 4, 5, 6\}$

## ◆ Binary extension field $\text{GF}(2^m)$

- Vector space of dimension  $m$  over  $\text{GF}(2)$
- **Polynomial basis**:  $B = \{1, \alpha, \alpha^2, \dots, \alpha^{m-1}\}$
- Elements: binary polynomials of degree  $m - 1$   
 $a(t) = a_{m-1}t^{m-1} + \dots + a_2t^2 + a_1t + a_0$  with  $a_i \in \{0, 1\}$
- Arithmetic modulo **irreducible polynomial**  $p(t)$
- E.g.:  $\text{GF}(2^3) = \{0, 1, t, t+1, t^2, t^2+1, t^2+t, t^2+t+1\}$

- ◆ **Software implementation**
  - Representation of long integers as **array of words**
  - Software **algorithms** operate on these words
  - Operand Scanning vs. **product scanning**
- ◆ **Hardware implementation**
  - Operands are stored in **long registers**
  - **Serial/parallel** architecture (e.g. bit-serial multiplier)
  - **Word-level** architecture:  $(w \times w)$ -bit multiplier
- ◆ **Reduction modulo  $p$** 
  - Very fast for **Mersenne-like primes**

## ◆ Addition in $GF(2^m)$

- Addition of coefficients modulo 2 (bit-wise XOR)

## ◆ Multiplication of binary polynomials

- Binary method: **Shift and XOR**

- Example:  $a(t) = t^3 + t + 1 = 1011$

$$b(t) = t^2 + 1 = 0101$$

- Product has degree of  $2m - 2$

$$\begin{array}{r}
 1011 \otimes 0101 \\
 \hline
 0000 \\
 1011 \\
 0000 \\
 1011 \\
 \hline
 0100111
 \end{array}$$

## ◆ Reduction modulo $p(t)$

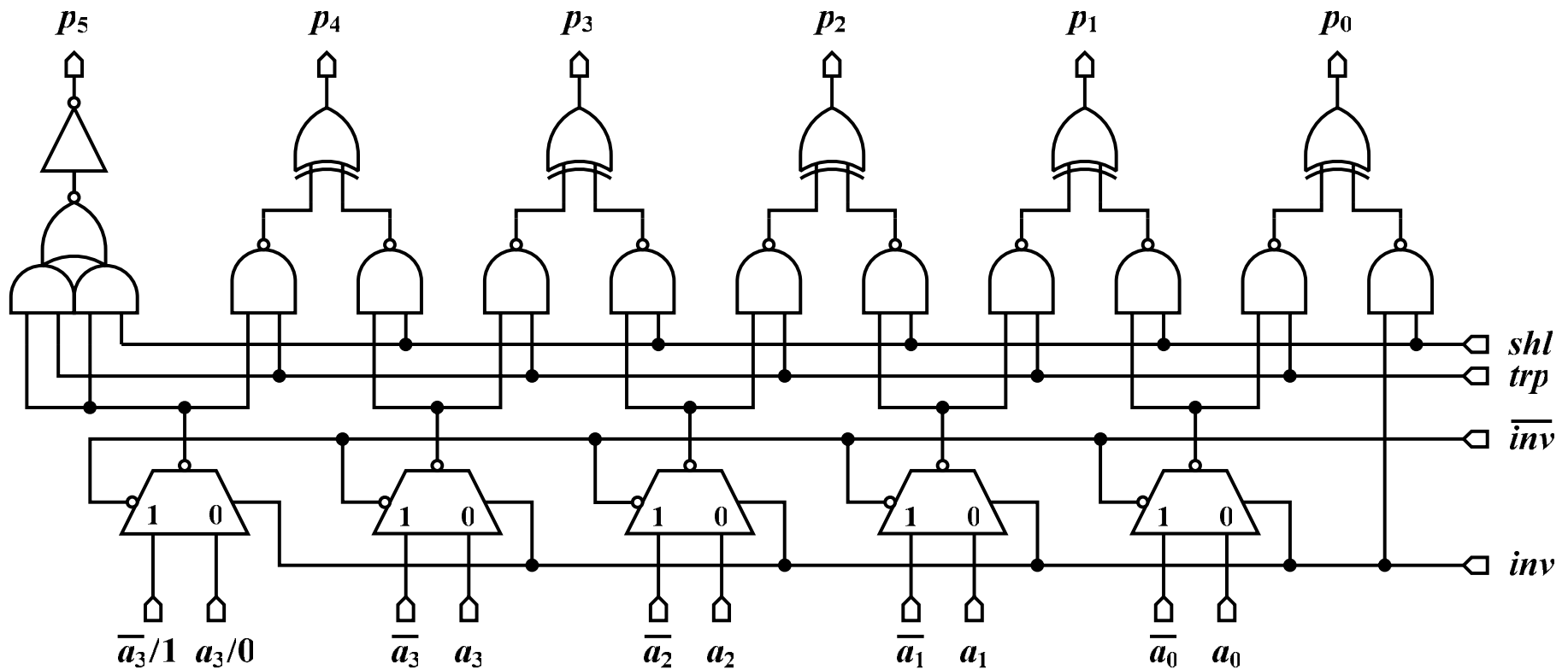
- Fast when  $p(t)$  is a trinomial or pentanomial

- ◆ **Generation of partial products**
  - Radix-2: simple logical AND
  - Radix-4: modified **Booth recoding** (MBR)
  - For binary polynomials: radix- $t$  and **radix- $t^2$**
- ◆ **Addition of partial products**
  - Array vs. tree architecture
  - INT: **redundant representation** (carry-save form)
  - Polynomials: **simple XOR**, no carry
- ◆ **Final adder**
  - Redundant-to-binary conversion (only for integers)

- ◆ **Unified multiplier datapath**
  - **Structural similarities** between INT and POLY mult.
  - Use the **same datapath** for both types of operands
- ◆ **Advantages**
  - Support for both  $GF(p)$  and  $GF(2^m)$
  - **Less hardware** than two separate multipliers
- ◆ **Unified (16 x 16)-bit multiplier + 40-bit accu**
  - **Functional unit** in an application-specific processor
  - **Arithmetic core** of a cryptographic co-processor

- ◆ **Unified radix-2/radix- $t$  multiplier**
  - **INT**: partial product is either 0 or  $A$
  - **POLY**: partial product is either 0 or  $a(t)$
  - Generation of partial products with **AND gates**
  - **16 partial products**
- ◆ **Unified radix-4/radix- $t^2$  multiplier**
  - **INT**: PP is from the set  $\{-2A, -A, 0, A, 2A\}$
  - **POLY**: PP is from the set  $\{0, a(t), t \cdot a(t), a(t) \oplus t \cdot a(t)\}$
  - Radix-4 PPG must be able to **invert, shift, and XOR**
  - **9 partial products** (or 8 for POLY mult.)

# Radix-4/Radix- $t^2$ PPG



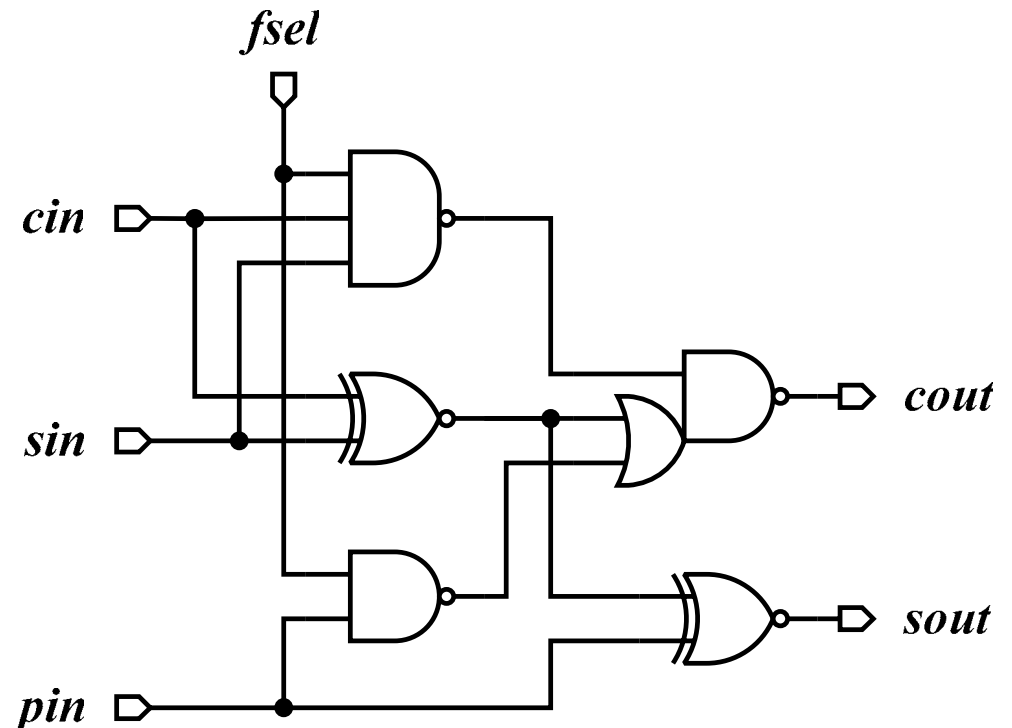
- Signals: *inv* (invert), *shl* (shift left), and *trp* (transport)
- MUX for inversion, NAND/XOR gates for shift and XOR

A **dual-field adder** (DFA) is a full adder capable of performing addition both with and without carry

**Integer mode:**  $fsel = 1$

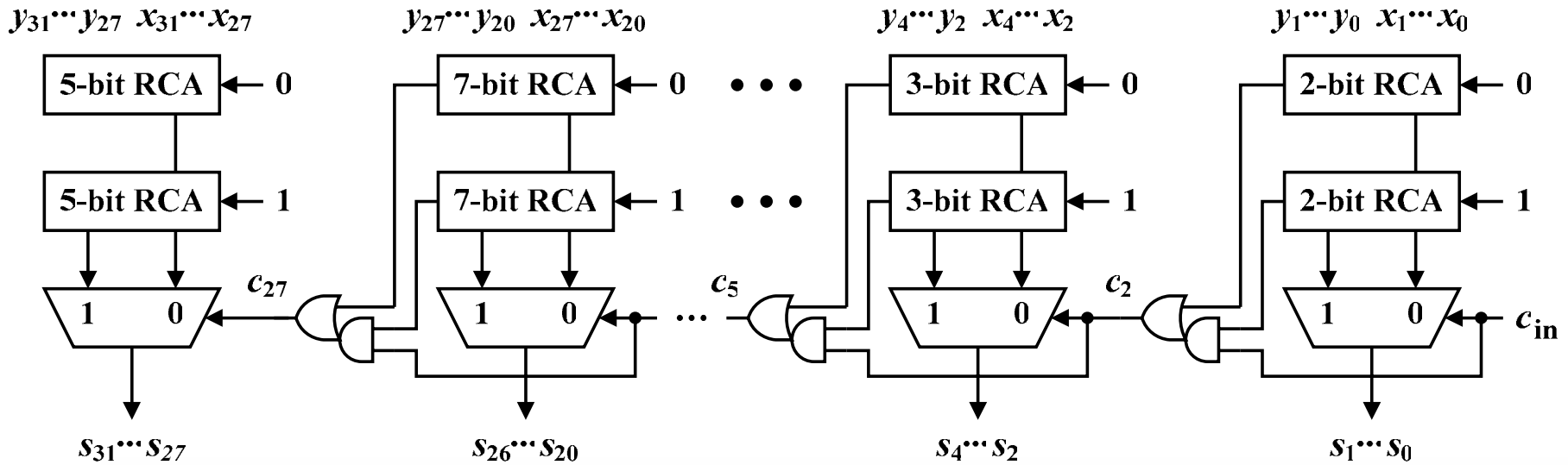
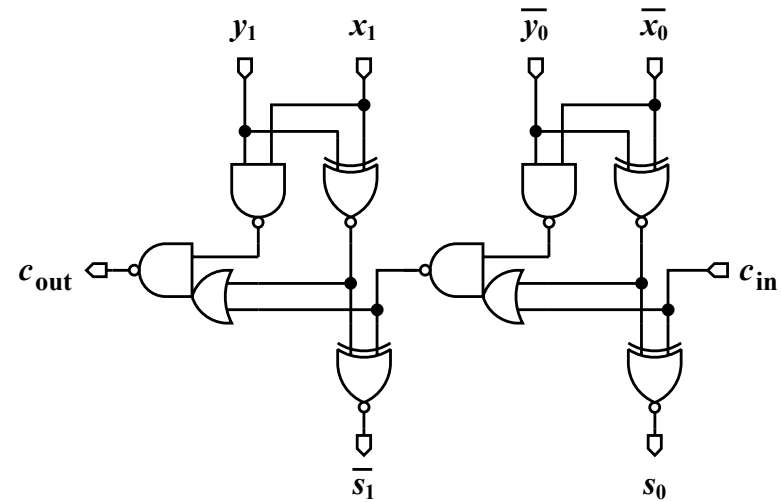
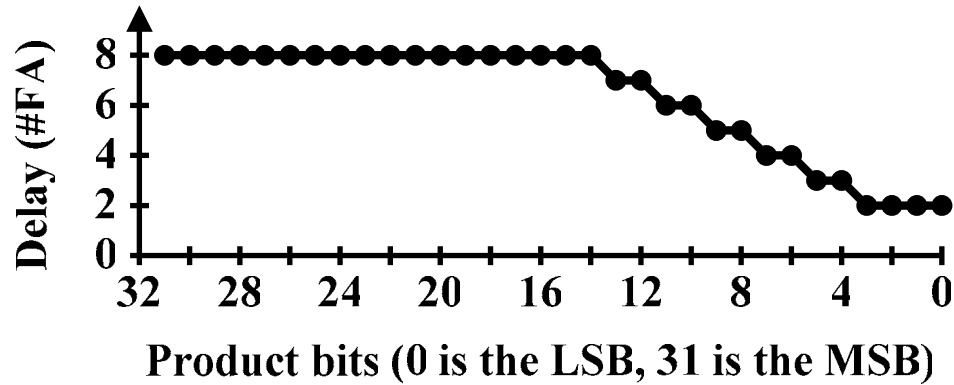
**Polynomial mode:**  $fsel = 0$

Only **slightly larger** than a conventional multiplier

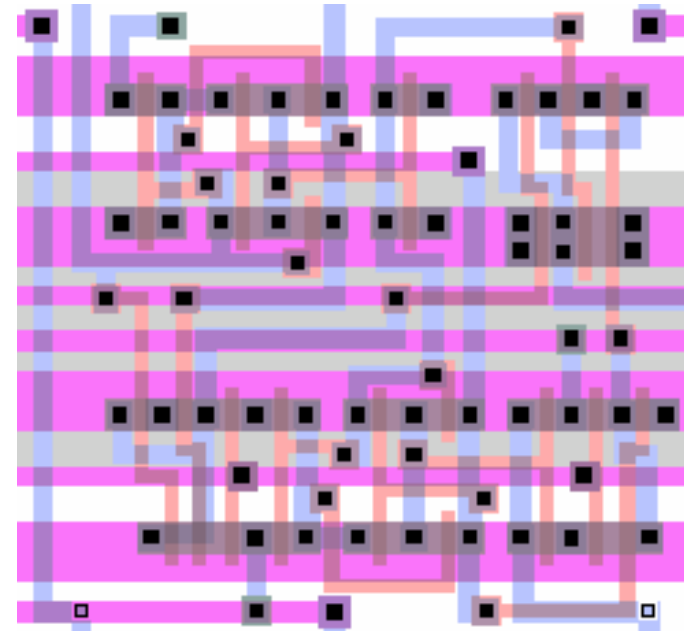


A properly designed unified multiplier consumes **35% less power** in POLY-mode than in INT-mode

# Final Adder



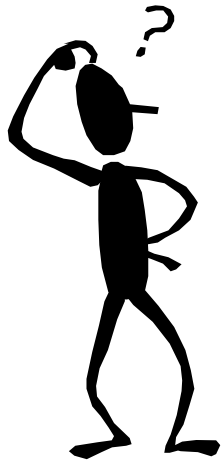
- ◆ **(16 x 16 + 40)-bit MAC unit**
  - Unified radix-2 multiplier
  - Unified radix-4 multiplier
- ◆ **Full-custom design**
  - Optimized for low power
  - 0.6  $\mu\text{m}$  CMOS technology
- ◆ **Unified radix-4 multiplier wins big**
  - Better power consumption, delay, PDP, and EDP than radix-2 multiplier
  - POLY more energy-efficient than INT



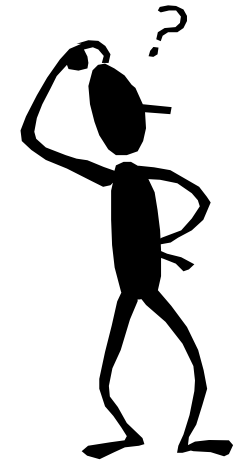
| Parameter                | Radix-2 version           | Radix-4 version           |
|--------------------------|---------------------------|---------------------------|
| Transistor count         | 12,384                    | 11,744                    |
| Delay (INT Mode)         | 82.4 nsec                 | 54.3 nsec                 |
| Avg. current (INT mode)  | 7.37 mA                   | 5.75 mA                   |
| PDP (INT mode)           | 2.43 nJ                   | 1.90 nJ                   |
| EDP (INT mode)           | $200.2 \cdot 10^{-18}$ Js | $103.0 \cdot 10^{-18}$ Js |
| Delay (POLY Mode)        | 66.8 nsec                 | 38.0 nsec                 |
| Avg. current (POLY mode) | 3.66 mA                   | 3.49 mA                   |
| PDP (POLY mode)          | 1.21 nJ                   | 1.15 nJ                   |
| EDP (POLY mode)          | $80.7 \cdot 10^{-18}$ Js  | $43.8 \cdot 10^{-18}$ Js  |

Results for  $f = 10$  MHz and  $V_{dd} = 3.3$  V

- ◆ **Unified (16 x 16 + 40)-bit MAC unit**
  - Same datapath for both types of operands
  - Execute MUL/MAC operations in 1 clock cycle
  - Array architecture
  - Both radix-2 and radix-4 is possible
  - Radix-4 multiplier better in terms of delay, power, and energy characteristics
- ◆ **POLY mode consumes less power**
  - No redundant representation (carries are zero)
  - 39% less power in poly mode (for radix-4 version)



## Questions ???



This research was supported by the Austrian Science Fund (FWF) under grant number P16952-N04 ("Instruction Set Extensions for Public-Key Cryptography"). The work described in this paper has been supported in part by the European Commission through the IST Programme under contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.