Department of Electronics and Multimedia Communications
Faculty of Electrical Engineering and Informatics
Technical University of Košice
Park Komenského 13, 04120 Košice, SLOVAKIA

Chair for Embedded Security
Ruhr-Universität Bochum
Universitätsstr. 150, 44801 Bochum
GERMANY

# A Very Lightweight Reconfigurable Elliptic Curve Crypto-Processor

**Michal Varchola**
**michal@varchola.com**

Tim Güneysu
gueneysu@crypto.rub.de

Oliver Mischke
mischke@crypto.rub.de

Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi)
June 15-18th 2011,  Bochum, Germany

# Outline

Related Work

Motivation

The MicroECC Requirements

Background

HW Implementation

Results & Comparison

Conclusion & Future Work

# Related Work

## Jo Vliegen et al., ASAP 2010

"A compact FPGA-based architecture for elliptic curve cryptography over prime fields"

Authors claimed:

"The comparison with other implementations on the same generation of FPGAs learns that our design occupies the smallest area."

Their ECC processor works over 256-bit prime fields for all elliptic curves of a special form

They use Montgomery multiplication

| **Datapath** | **Slices** | **HW mults.** | **BRAMs** | **ECPM** | **Family** |
|---|---|---|---|---|---|
| 16-bit | 1694 | 2 | 9 | 29.83ms | Virtex-II Pro |
| 32-bit | 1947 | 9 | 9 | 15.75ms | Virtex-II Pro |

# Related Work

## Tim Guneysu and Christof Paar, CHES 2008

“Ultra High Performance ECC over NIST Primes on Commercial FPGAs”

Highly parallel architecture with wide use of DSPs

NIST-224 and NIST-256 elliptic curves support

Integer Multiplication + Fast reduction algorithm

| Curve | Slices | DSPs | BRAMs | ECPM | Family |
|---|---|---|---|---|---|
| NIST-224 | 1580 | 26 | 11 | 365us | Virtex-IV |
| NIST-256 | 1715 | 32 | 11 | 495us | Virtex-IV |
| NIST-224 | 24452 | 468 | 198 | 26.5us | Virtex-IV |
| NIST-256 | 24574 | 512 | 176 | 40.5us | Virtex-IV |

# Motivation

## Why we need another FPGA ECC processor?

- ECC is favorite asymmetric key cryptosystem for small embedded platforms

- FPGAs are preffered platform for such cryptosystems

- Existing fast ECC processors requires plenty of logic resources or dedicated DSP blocks

- DSP blocks often disable multi-vendor support

- Existing Lightweight designs are quite slow

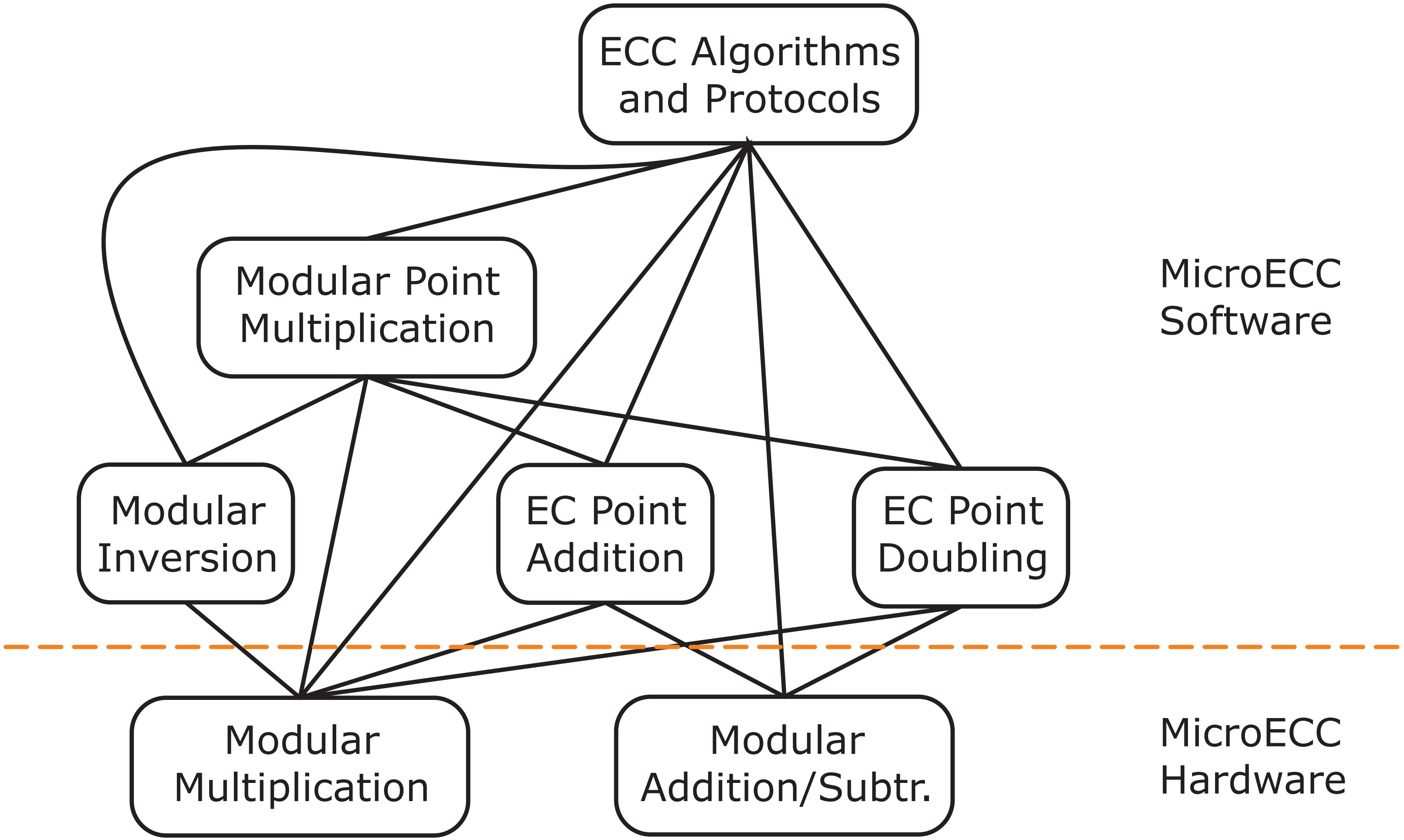
- To implement power analysis countermesasures

# The MicroECC Requirements

- Small footprint in FPGA (important for the Actel)
- As good performance as possible
- Multi-vendor support (Actel, Xilinx, ... )
- Side channel analysis countermeasures
- Reprogrammable NIST-224 or NIST-256 support
- Embedded software Compiler & Simulator

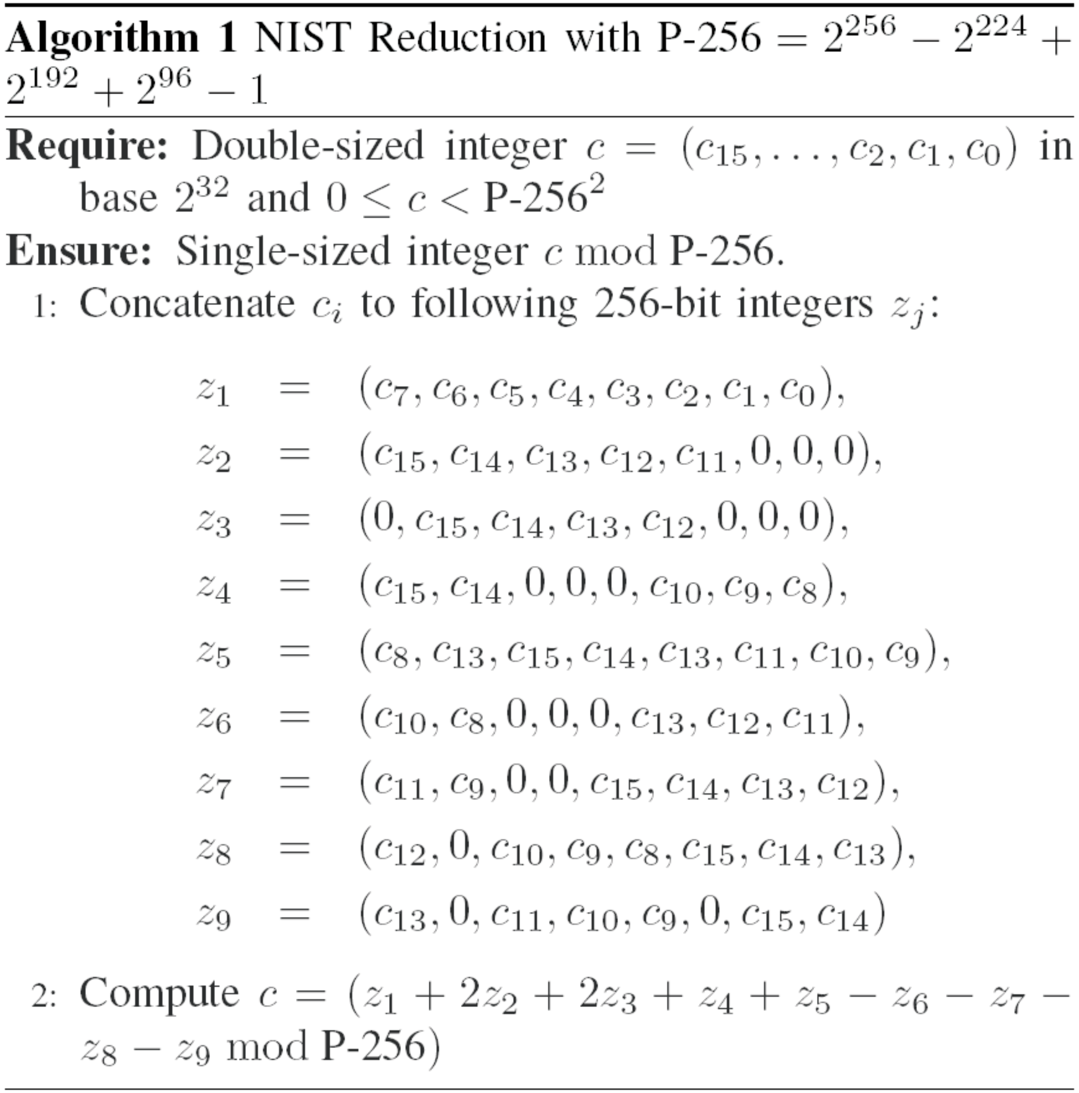
# Background

## The ECC Stack and the MicroECC



# Background

## Modular Multiplication with Fast Reduction



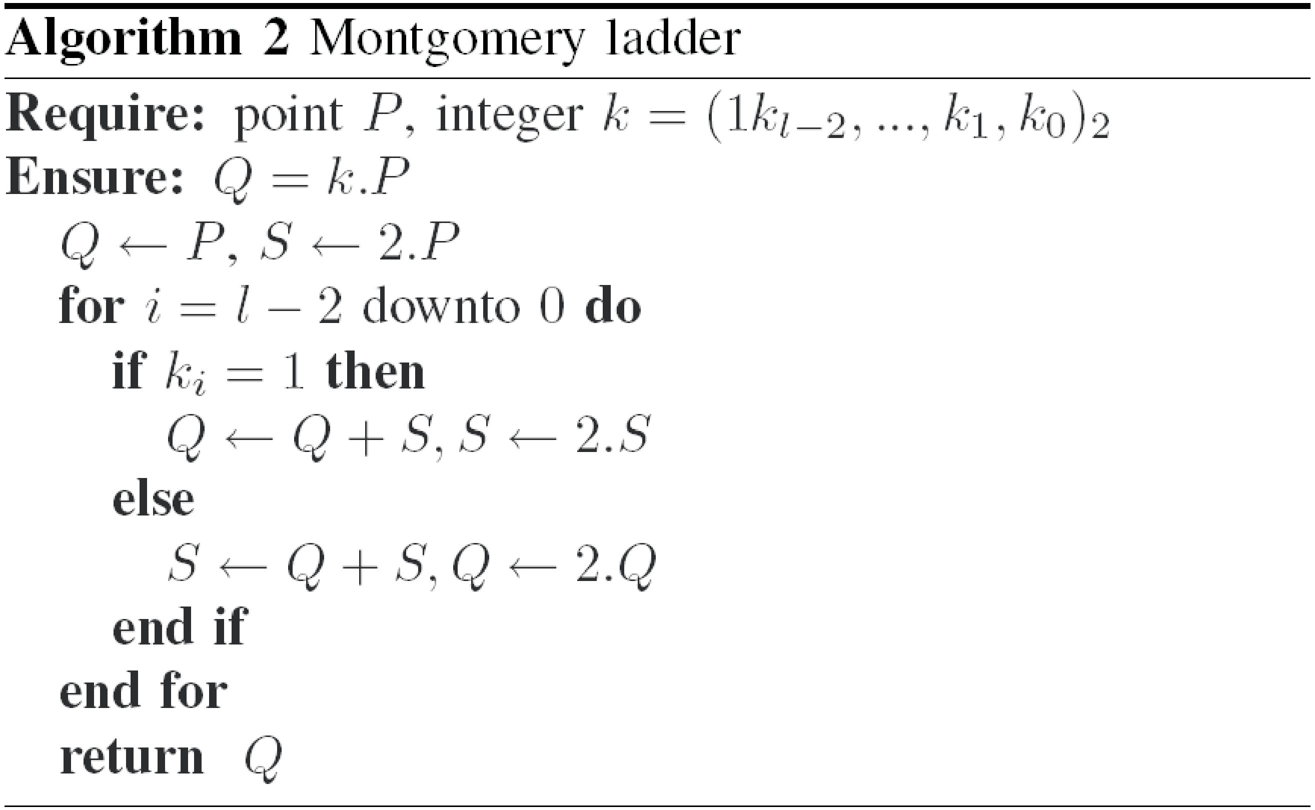
**Algorithm 1** NIST Reduction with P-256 $= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$

```
Require: Double-sized integer $c = (c_{15}, \ldots, c_2, c_1, c_0)$ in base $2^{32}$ and $0 \leq c <$ P-256$^2$
Ensure: Single-sized integer $c$ mod P-256.
Concatenate $c_i$ to following 256-bit integers $z_j$:
    $z_1 = (c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$,
    $z_2 = (c_{15}, c_{14}, c_{13}, c_{12}, c_{11}, 0, 0, 0)$,
    $z_3 = (0, c_{15}, c_{14}, c_{13}, c_{12}, 0, 0, 0)$,
    $z_4 = (c_{15}, c_{14}, 0, 0, 0, c_{10}, c_9, c_8)$,
    $z_5 = (c_8, c_{13}, c_{15}, c_{14}, c_{13}, c_{11}, c_{10}, c_9)$,
    $z_6 = (c_{10}, c_8, 0, 0, 0, c_{13}, c_{12}, c_{11})$,
    $z_7 = (c_{11}, c_9, 0, 0, c_{15}, c_{14}, c_{13}, c_{12})$,
    $z_8 = (c_{12}, 0, c_{10}, c_9, c_8, c_{15}, c_{14}, c_{13})$,
    $z_9 = (c_{13}, 0, c_{11}, c_{10}, c_9, 0, c_{15}, c_{14})$
Compute $c = (z_1 + 2z_2 + 2z_3 + z_4 + z_5 - z_6 - z_7 - z_8 - z_9$ mod P-256)
```

# Background

## SCA Countermeasures

**SPA:**



**Algorithm 2** Montgomery ladder

```
Require: point $P$, integer $k = (1k_{l-2}, ..., k_1, k_0)_2$
Ensure: $Q = k.P$
  $Q \leftarrow P$, $S \leftarrow 2.P$
  for $i = l - 2$ downto 0 do
    if $k_i = 1$ then
      $Q \leftarrow Q + S, S \leftarrow 2.S$
    else
      $S \leftarrow Q + S, Q \leftarrow 2.Q$
    end if
  end for
  return $Q$
```

**DPA:**

Randomizing the homogeneous projective coordinates(X,Y,Z) with a random λ ≠ 0 to (λX, λY, λZ). The random variable λ can be updated in every execution or after each doubling or addition.

# HW Implementation

## MicroECC-1 and MicroECC-2

**MicroECC-1:**

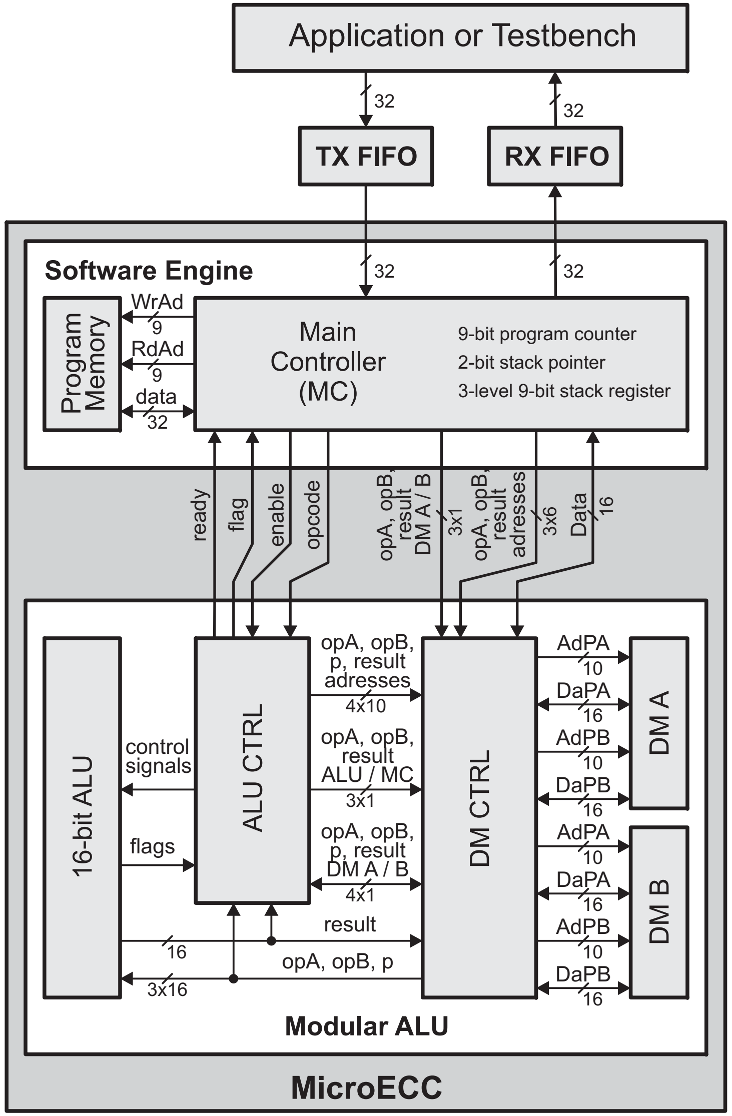
- 16-bit datapath
- NIST-256 curve support
- Virtex-II Pro FPGA
- Evaluated in HW

**MicroECC-2:**

- 16-bit or 32-bit datapath
- NIST-224 or NIST-256 curve support
- Xilinx and Actel FPGA support
- Highly optimized arithmetic unit
- In development now

# HW Implementation

## MicroECC Architecture



- Software Engine executes the embedded program stored in Program Memory
- Arithmetic and Logic Unit (ALU) has 16-bit (or 32-bit) datapath
- ALU CTRL is set of FSMs which controls execution of each instruction
- Data Memory Controller (DM CTRL) controls reads and writes to DMs according decoded instruction
- Data Memories (DM) store constants variables and instructions for the fast reduction algorithm
- Two dual port data memories DM A and DM B have 4 data ports: we can read 3 operands and write 1 result

# HW Implementation

## MicroECC Instruction Set

Instructions related to flow of the embedded program:

**WRPGM** - Write to Program Memory of a custom MC
**RDPGM** - Read from Program Memory of a custom MC
**EXERTN** - Execute Routine
**JMPFT** - Jump if Flag is True, otherwise continue
**JMPFF** - Jump if Flag is False, otherwise continue
**JMP** - Unconditional Jump
**CALL** - Routine Call; maximum three consecutive calls due to 3-level hardware stack
**RET** - Return from Routine

# HW Implementation

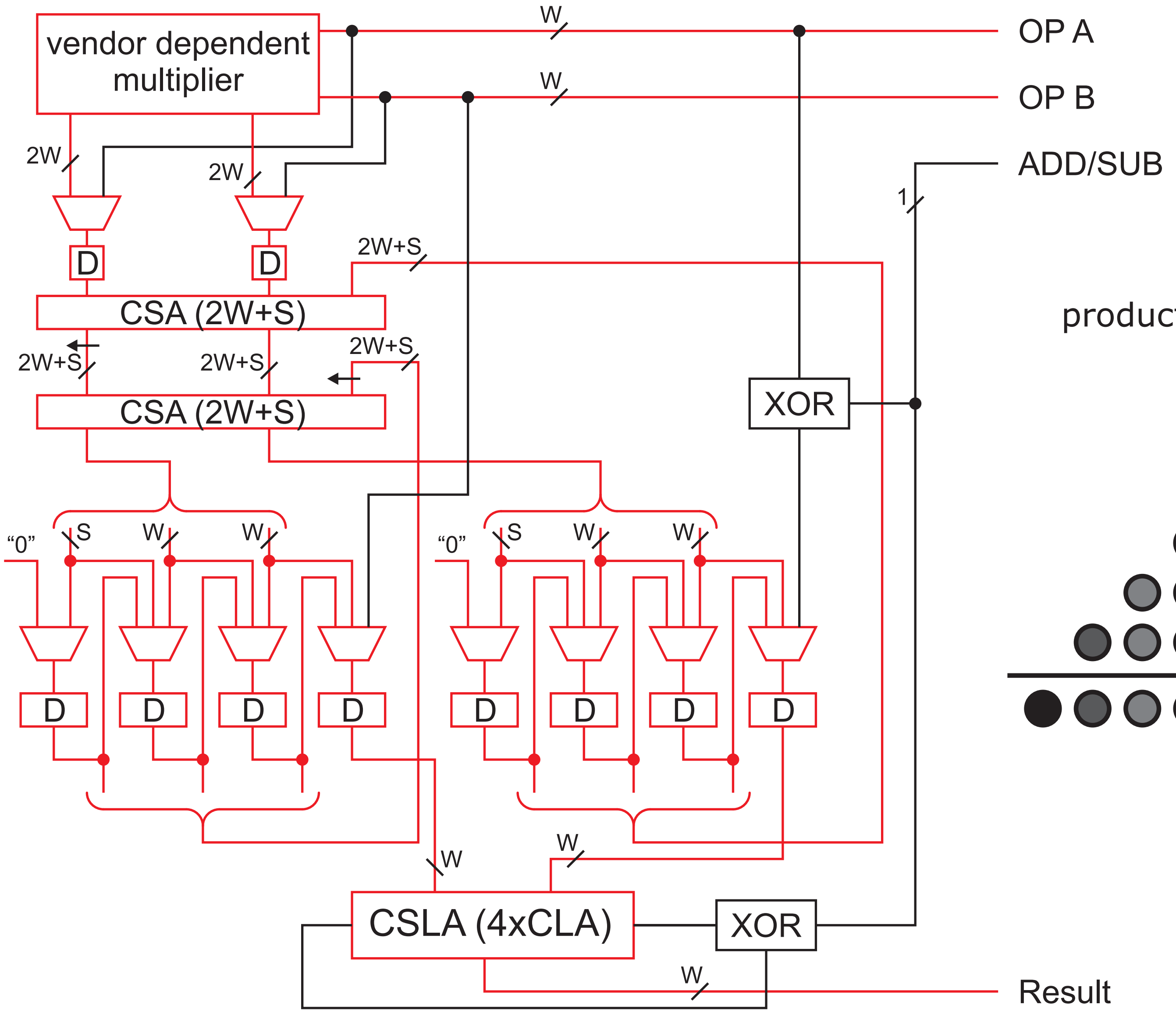
## MicroECC Instruction Set

Instruction related to the Arithmetic and Logic Unit:

**CHKB** - Check Bit
**WRITE** - Write to Register File (DM A and DM B)
**READ** - Read from Register File (DM A and DM B)
**MOVE** - Move data in Register File
**MADD** - Modular Addition
**MSUB** - Modular Subtraction
**MMUL** - Modular Multiplication
**CMPGR** - Comparison: Is A Grater than B?
**CMPEQ** - Comparison: Are A and B Equal?
**CMPLO** - Comparison: Is A Lower than B?

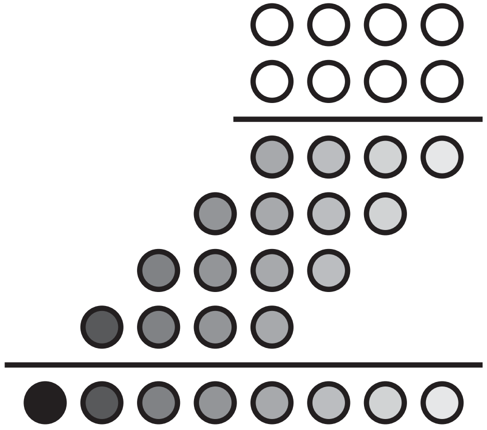
# HW Implementation

## Arithmetic & Logic Unit - Integer Product



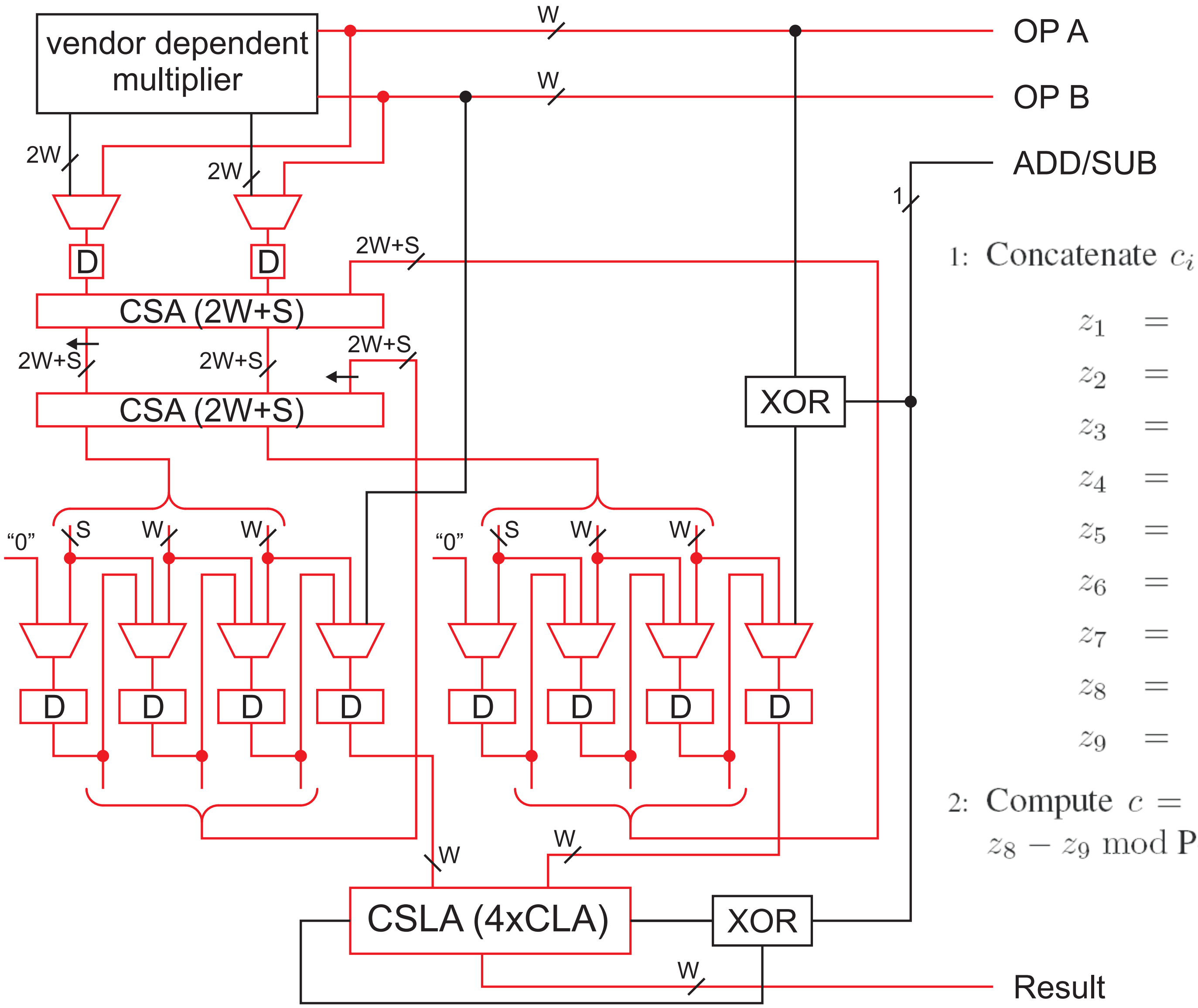
16-bit datapath: W=16
32-bit datapath: W=32
S=4

product scanning method:



# HW Implementation

## Arithmetic & Logic Unit - Fast Reduction



16-bit datapath: W=16
32-bit datapath: W=32
S=4

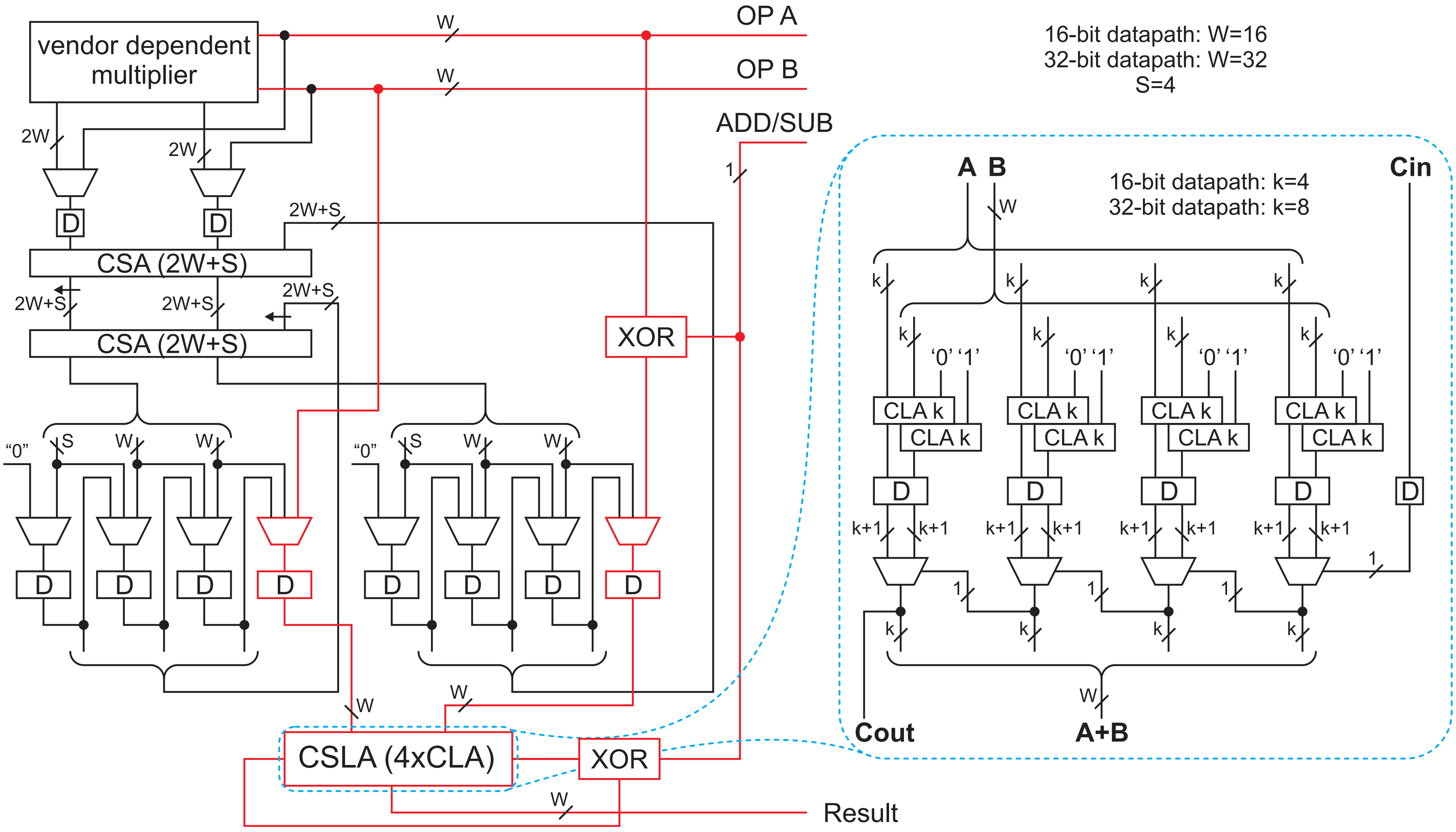
1: Concatenate $c_i$ to following 256-bit integers $z_j$:

$$
\begin{aligned}
z_1 &= (c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0),\\
z_2 &= (c_{15}, c_{14}, c_{13}, c_{12}, c_{11}, 0, 0, 0),\\
z_3 &= (0, c_{15}, c_{14}, c_{13}, c_{12}, 0, 0, 0),\\
z_4 &= (c_{15}, c_{14}, 0, 0, 0, c_{10}, c_9, c_8),\\
z_5 &= (c_8, c_{13}, c_{15}, c_{14}, c_{13}, c_{11}, c_{10}, c_9),\\
z_6 &= (c_{10}, c_8, 0, 0, 0, c_{13}, c_{12}, c_{11}),\\
z_7 &= (c_{11}, c_9, 0, 0, c_{15}, c_{14}, c_{13}, c_{12}),\\
z_8 &= (c_{12}, 0, c_{10}, c_9, c_8, c_{15}, c_{14}, c_{13}),\\
z_9 &= (c_{13}, 0, c_{11}, c_{10}, c_9, 0, c_{15}, c_{14})
\end{aligned}
$$

2: Compute $c = (z_1 + 2z_2 + 2z_3 + z_4 + z_5 - z_6 - z_7 - z_8 - z_9 \bmod \text{P-256})$

# HW Implementation

## Arithmetic & Logic Unit - Modular Addition/Subtraction



# HW Implementation

## Vendor Dependent Multiplication Block

**Xilinx:**

MicroECC-2/16: one 16x16 HW Multiplier block

MicroECC-2/32: four 16x16 HW Multiplier blocks and CSA

**Actel:**

MicroECC-2/16: parallel 16x16 Multiplier (1100 tiles)

MicroECC-2/32: parallel 32x32 Multiplier (4000 tiles)

both have two outputs after the CSA addition

# Results & Comparison

## MicroECC-1/2 vs. other designs

| Design | FPGA | Slices<br>Tiles | HW<br>mult | RAM<br>blocks | Max.<br>Freq. | ECPM | Curve |
|---|---|---|---|---|---|---|---|
| MicroECC-1/16 | V2P | 615 | 1 | 3 | **165 MHz** | **15 ms** | N-256 |
| MicroECC-1/16 | SmFus | 3720 | 0 | 12 | **55 MHz** | **45 ms** | N-256 |
| MicroECC-2/16 | V2P | 650* | 1 | 3 | **265 MHz*** | **9 ms*** | N-256 |
| MicroECC-2/32 | V2P | 1100* | 4 | 3 | **225 MHz*** | **3.5 ms*** | N-256 |
| MicroECC-2/16 | SmFus | 3600* | 0 | 8* | **150 MHz*** | **16 ms*** | N-256 |
| MicroECC-2/32 | SmFus | 7000* | 0 | 8* | **120 MHz*** | **7 ms*** | N-256 |
| Vliegen 16 | V2P | 1694 | 2 | 9 | **108 MHz** | **30 ms** | any 256 |
| Vliegen 32 | V2P | 1947 | 7 | 9 | **68 MHz** | **15 ms** | any 256 |
| McIvor et al. | V2P | 15755 | 256 | 0 | **39.5 MHz** | **3.84 ms** | any 256 |
| Orlando et al. | V1000 | 11416 | 0 | 35 | **40 MHz** | **3 ms** | N-192 |

V2P - Xilinx Virtex-II Pro
SmFus - Actel Smart Fusion
V1000 - Xilinx XCV1000E-8

* estimated values according current state of development

# Conclusion & Future Work

**Conclusion:**

- Better performance and less resources usage than Vliegen et al. design
- Just NIST prime curves support (NIST-224,256)
- Compiler and Simulator written in ANSI C
- All NIST curve parameters are held in DM including instructions for fast reduction; in other words: selection between NIST-224 or NIST-256 is done on a software level, i.e. dynamicaly switchable

**Future work:**

- Completing the MicroECC-2 processor with the goal of the estimated performance
- SPA and DPA evaluation

# Thank You