A Reconfigurable Arithmetic Logic Unit for Elliptic Curve Cryptosystems over $GF(2^m)$

The 46th IEEE International Midwest Symposium
On Circuits and Systems
December 27th-30th, 2003
Cairo, Egypt

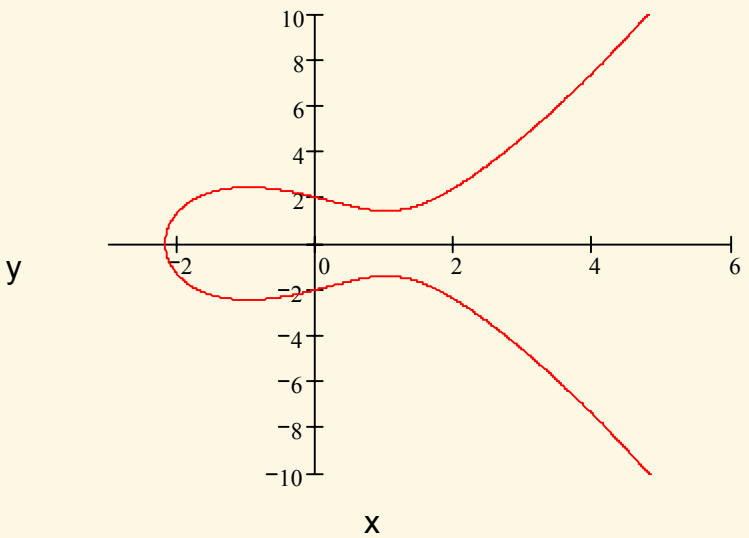
Mathias Schmalisch · Dirk Timmermann

Contents

Motivation

- Elliptic Curve Cryptography (ECC) uses smaller key length than other public key algorithms at the same level of security
  - ECC uses today 160 bit
  - RSA uses today 1024 bit
- ECC can be used for authentication, encryption, digital signatures, key exchange and so forth
- There exist several hardware implementations for ECC, but this implementations uses algorithms with few inversions
- Other algorithms with more inversions are better, but you need a fast implementation for the inversion
- This presentation shows a ALU for ECC with a fast hardware inversion and compares it with other hardware implementations

Elliptic Curves



- Weierstrass equation:

  $E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$

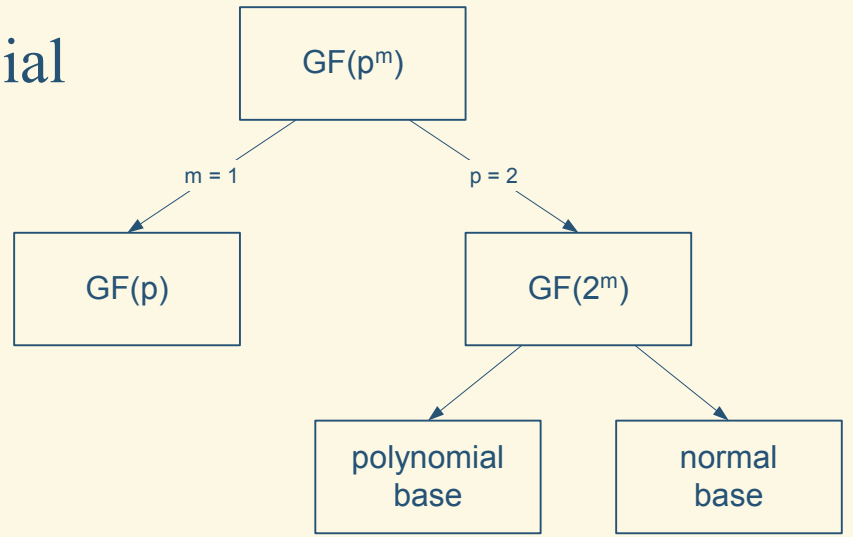
  $x, y, a_i \in GF(q)$

- E is the set of solutions of the Weierstrass equation in the affine plane
- Elliptic curves used in cryptography will be mapped on finite fields $GF(q)$
- The curve may not be singular
  - Discriminant $\Delta \neq 0$
  - Otherwise the curve has edges or is not continuous

Finite Fields $GF(q)$

- For finite fields there exist two possible notations
  - $GF(q) = \boldsymbol{F}_q$
- Where $q = p^m$ with p a prime number and m a nonnegative integer
- For cryptography there only the special cases $m = 1$ or $p = 2$ are from interest
- For hardware computation the special case $p = 2$ is very interesting
  - char($GF(2^m)$) = 2

    $E: y^2 + xy = x^3 + ax^2 + b$



Point Operations on Elliptic Curves in $GF(2^m)$

- Point addition $P \neq Q$
- A line through P and Q

$$\lambda = (x_1 + x_2)/(y_1 + y_2)$$

- Point doubling $P = Q$
- A tangent at P

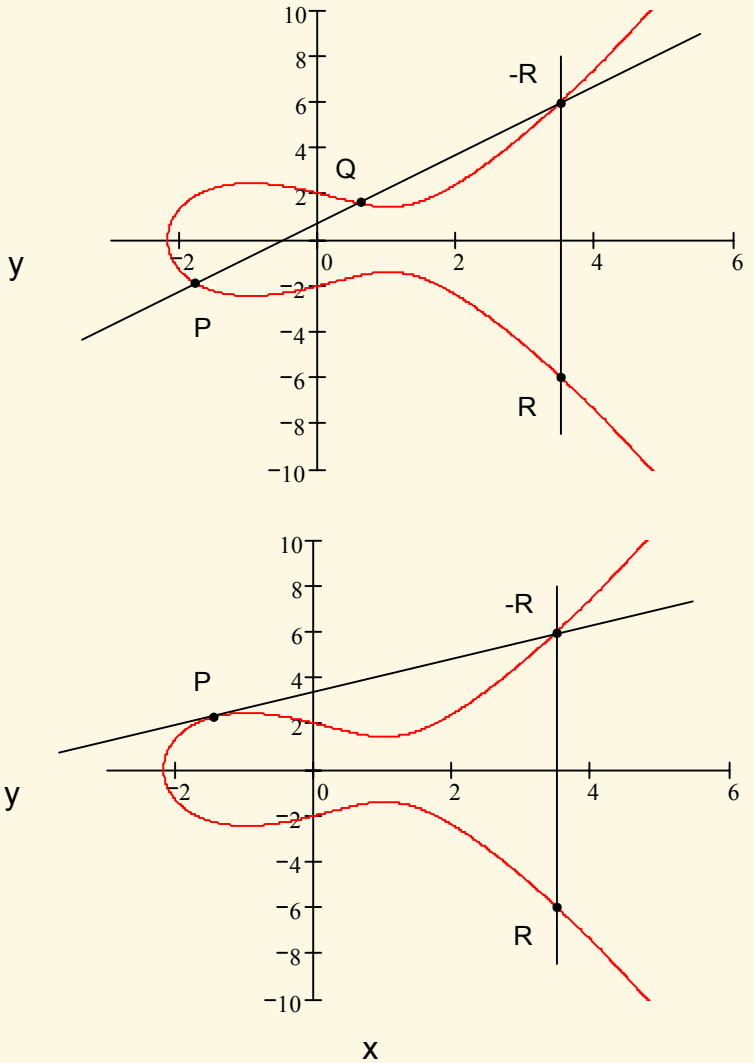
$$\lambda = x_1 + y_1/x_1$$

- Computing the coordinates of R

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \qquad \text{if } P \neq Q$$

$$y_3 = \lambda x_3 + x_3 + x_1^2 \qquad \text{if } P = Q$$



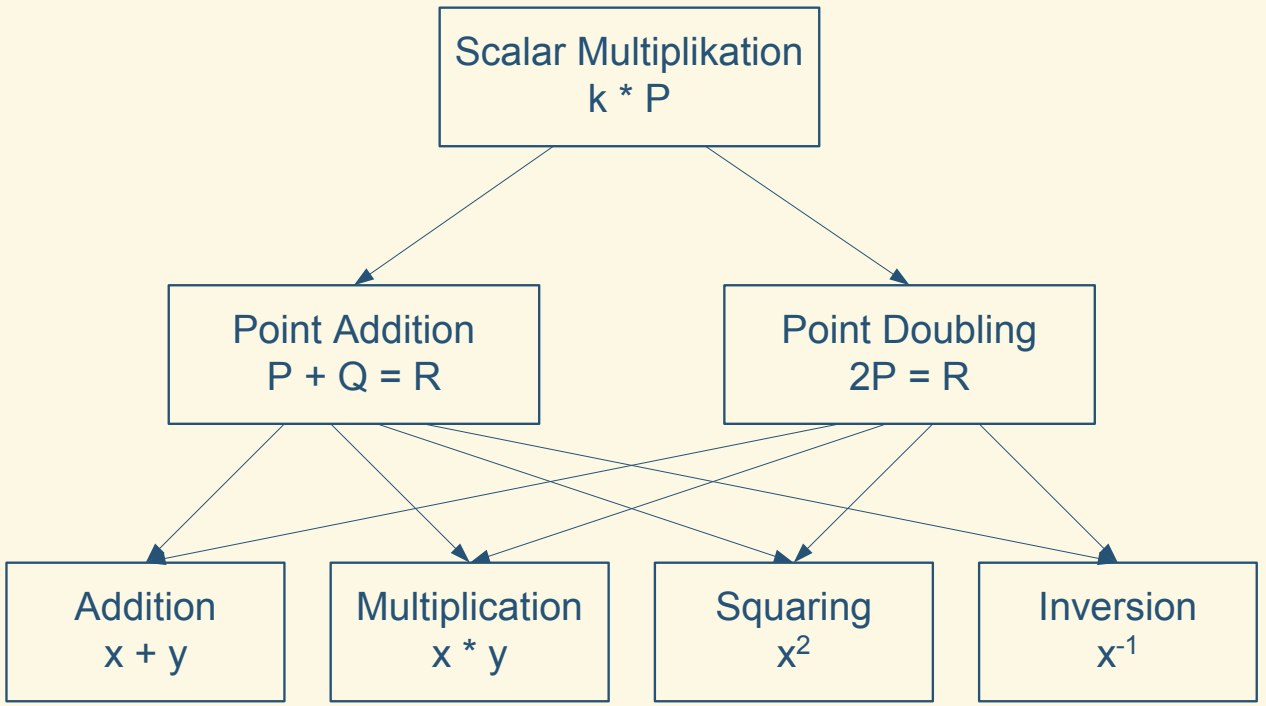
Finite Field Operations in $GF(2^m)$

- Needed mathematical operations in the finite field $GF(2^m)$
  - Addition, Multiplication, Division
- Squaring is a special case of the multiplication
  - But can be faster computed then a multiplication
- Division is very hard to compute, therefore the multiplication with the multiplicative inverses will be used

$$a / b = a * b^{-1}$$

- Final operations for the finite field $GF(2^m)$
  - Addition, Multiplication, Squaring, Inversion

Scalar Multiplication *k*P*

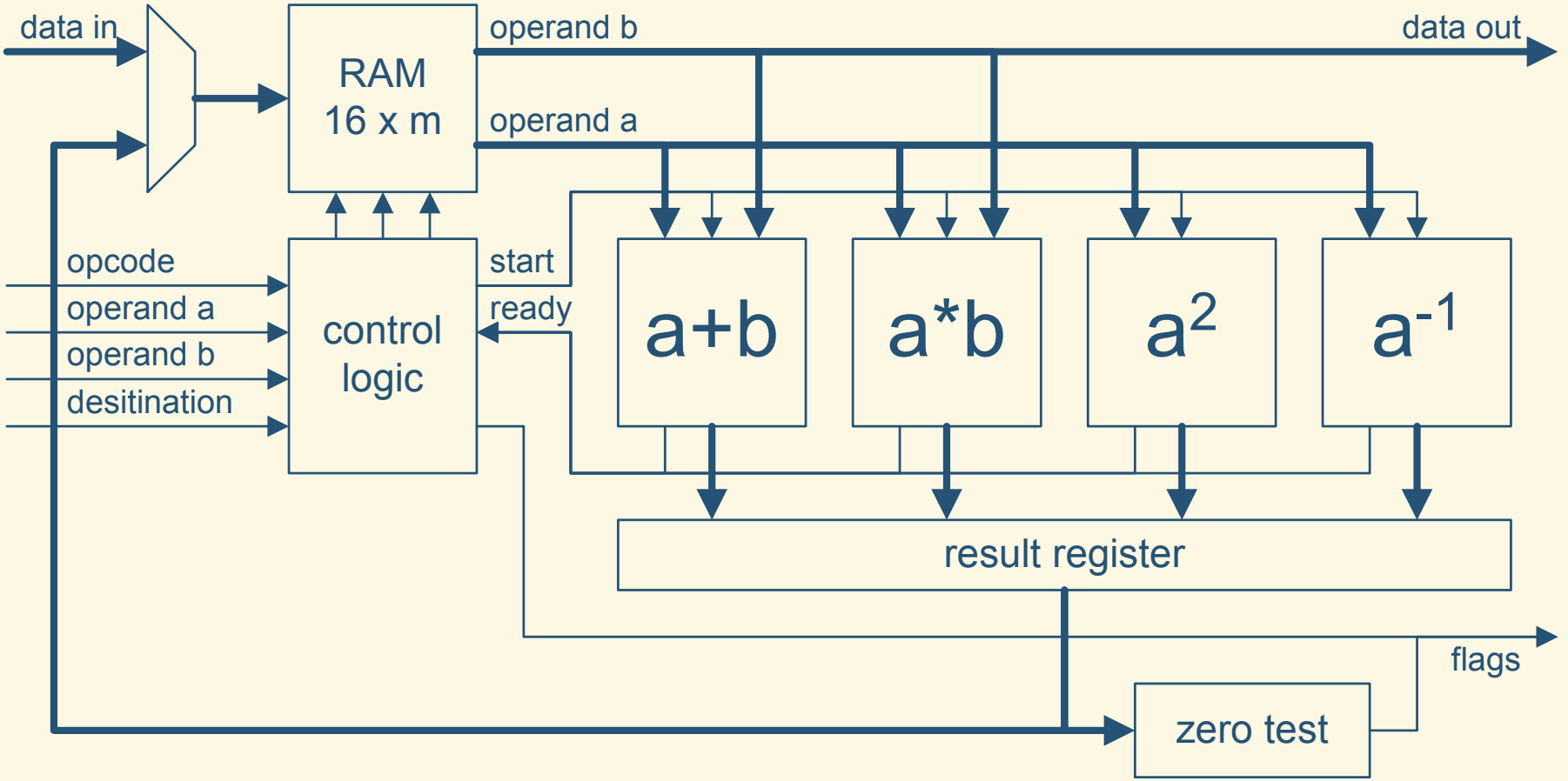


Scalar Multiplikation

Point Operations on Ellitpc Curve

Finite Field Operations

Structure of the ALU



Addition

- Addition in the finite field $GF(2^m)$
- Base function for the multiplication
- In the finite field there exist only the elements 0 and 1

$$A + A = 2A \bmod 2 = 0A = 0$$

- Addition of two bits without carry
- Therefore it is a XOR combination of two numbers A and B

- Fast computation, needs only one clock cycle

Multiplication

- Multiplication of A und B, each with a bit length of m
- The bit length of the result is $(2m - 1)$
- Reduction to bit length of m with the irreducible polynomial

- Serial multiplication with the paper and pencil method
- Summation of partial products

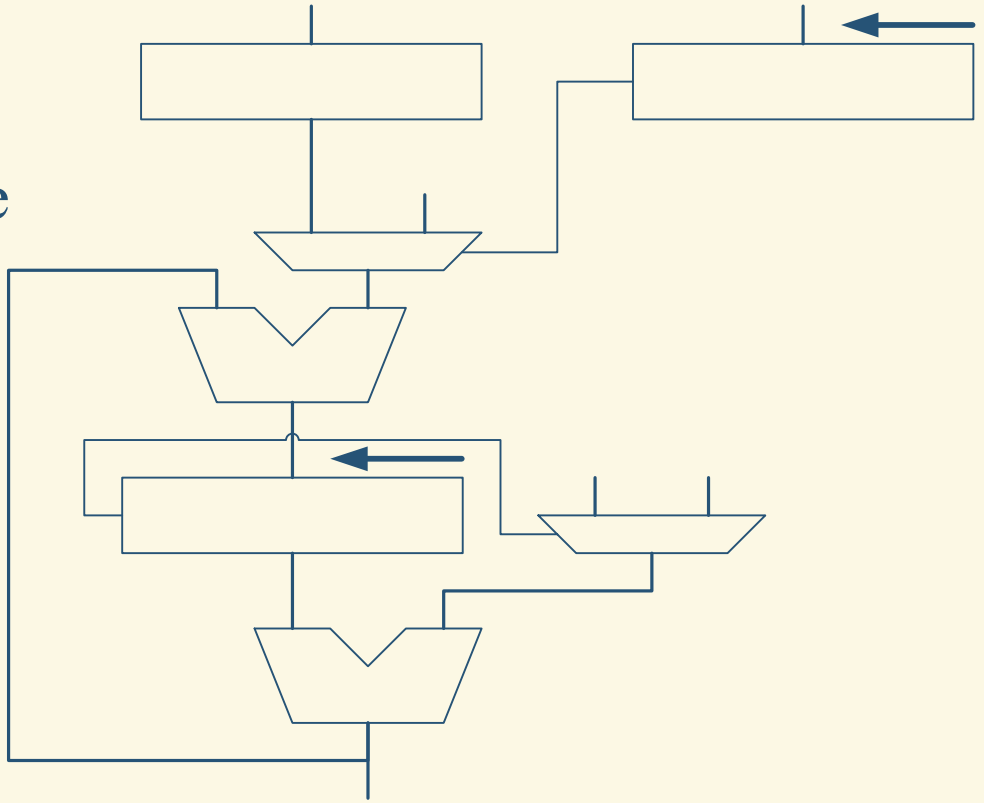
$$X = A \cdot B = \sum_{i=0}^{m} A \cdot b_i 2^i \qquad B = \left(b_m, \ldots, b_0\right)_2$$

Multiplication

- Well known architecture
  - But addition = XOR combination

- Including the reduction with the irreducible polynomial
- Therefore the result needs no additional reduction

- This serial architecture needs m clock cycles

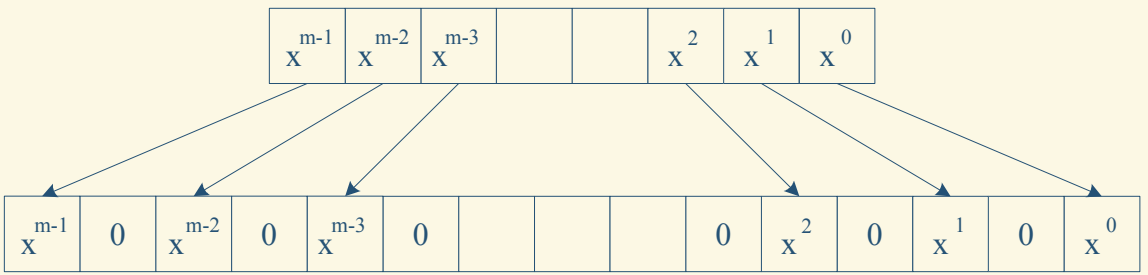


Squaring

- Squaring is a special case, then

$$(a+b)^2 = a^2 + 2ab + b^2 = a^2 + b^2$$

- Result can be computed immediately
  - But it has a bit length of $(2m - 1)$

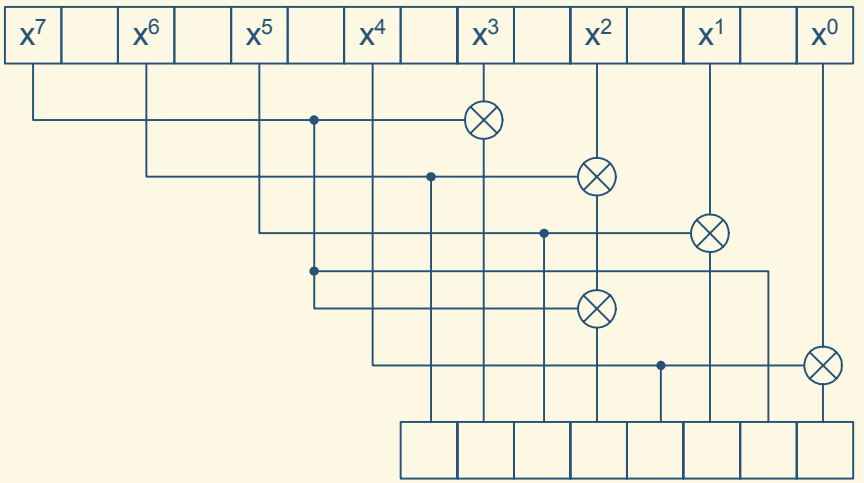


- A reduction with the irreducible polynomial is needed

Squaring

- Reduction of the result can be computed parallel

Example $GF(2^8)$: $x^8 + x^3 + x^0$



- Needs only one clock cycle
- If the irreducible polynomial is a trinomial, then the reduction needs few XOR combinations

- Only $(m + k - 1) / 2$ XOR combinations
  - Trinomial: $x^m + x^k + 1$
  - For our ALU Implementation
    $(167 + 6 - 1) / 2 = 86$ XOR's

Inversion

- There exist two possible methods to compute the multiplicative inverses
  - Euclid's Algorithm
  - Fermat's Theorem

- The input for the inversion must not be zero because

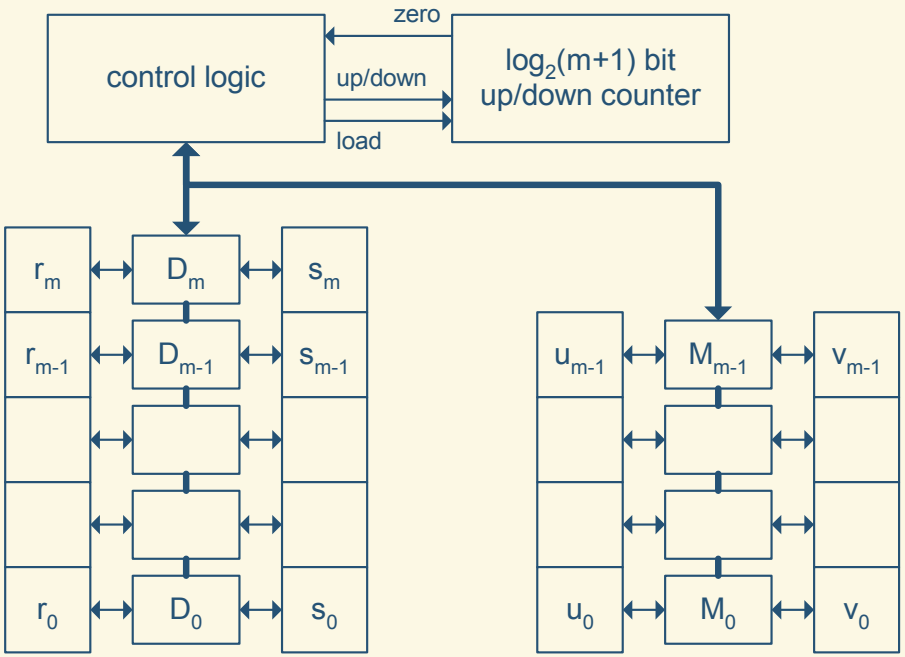
$$a \cdot a^{-1} = 1$$

- Therefore the zero test is needed

- For hardware computation the Euclidian Algorithm is the best choice, then for this algorithm there exist a good solution

Inversion

- Hardware solution of H. Brunner, A. Curiger und M. Hofstetter
- Inversion for finite fields $GF(2^m)$ on polynomial base
- Smallest known hardware solution for the inversion
- Computes the multiplicative inverses in m clock cycles
- It is possible to parallelize this hardware for faster computation



Our Implementation and Results

- ALU for finite fields $GF(2^{167})$ on polynomial base
  - Irreducible polynomial: $x^{167} + x^6 + 1$
- Developed in VHDL
- Configurable in field width and irreducible polynomial
- Synthesized with Synplify Pro 7.3 from Syplicity and Xilinx ISE 5.2i

- Synthesized on a Xilinx FPGA XC400E-8
  - Clock frequency: 82.3 MHz
  - Flip Flops: 1393
  - Equivalent gate count: 56936
  - Time for a scalar multiplication: 1.3 ms

Comparing the Results

Implementation Authors, Year	Target Platform	Field width	$k \cdot P$ (ms)	Throughput (kbit/s)
Okada, Torii, Itoh, Takenaka, 2000	Altera FPGA EPF10K, 3 MHz	163	80.7	2.0
Leung, Ma, Wong, Leong, 2000	Xilinx FPGA XCV300, 45 MHz	113	3.7	29.8
Ernst, Jung, Madlener, Huss, Blümel, 2003	Atmel FPSLIC AT94K40, 12 MHz	113	1.4	78.8
Ernst, Klupsch, Hauck, Huss, 2001	Xilinx FPGA XC4085XLA, 37 MHz	155	1.3	116.4
Orlando, Paar, 1999	Xilinx FPGA XCV400E, 76.7 MHz	167	0.21	776.7
Our ALU implementation	**Xilinx FPGA XCV400E, 82.3 MHz**	**167**	**1.3**	**126.2**

Conclusion

- Short introduction in the elliptic curve cryptography
- I Present the structure and implementation of our ALU
- We implemented an ALU with a serial multiplication and inversion
- Because of the hardware inversion we could use other algorithms and so it was possible to reach a high performance
- It is possible to parallelize the multiplication and inversion to reach higher performance