Eine ALU für die schnelle Berechnung der Kryptographie auf Basis elliptischer Kurven

Mathias Schmalisch · Marc-Sebastian Fiedler · Dirk Timmermann

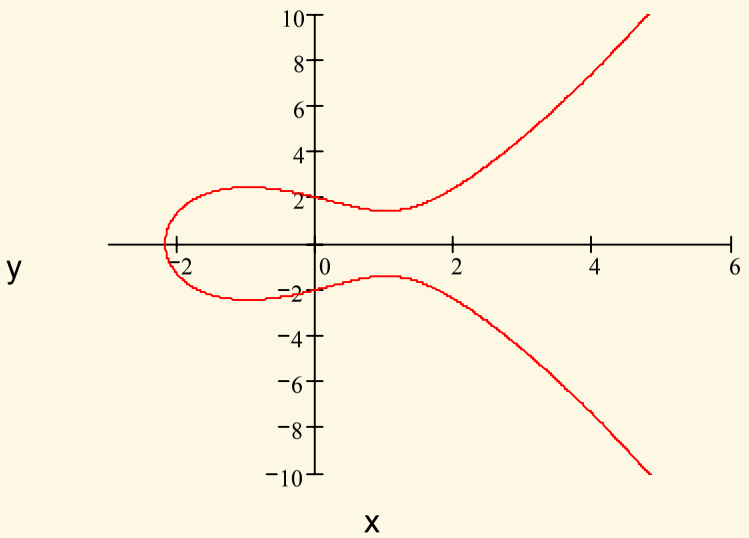
Inhalt

- Motivation
- Kryptographie auf Basis elliptischer Kurven
- Aufbau der ALU
- Funktionsweise der Rechenoperationen
- Ergebnisse und Ausblick
- Zusammenfassung

Motivation

- Kryptographie auf Basis elliptischer Kurven kommt mit wesentlich kleinerer Schlüsselbreite aus als andere Verfahren
- Hardwareberechnung ist wesentlich schneller als Softwareberechnung
- Bisher gibt es noch keine Hardware mit integrierter Invertierung
- Dadurch sind andere Algorithmen möglich, die bis zu 3 mal schneller sind

Elliptische Kurven



- Weierstrass Normalform:

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

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

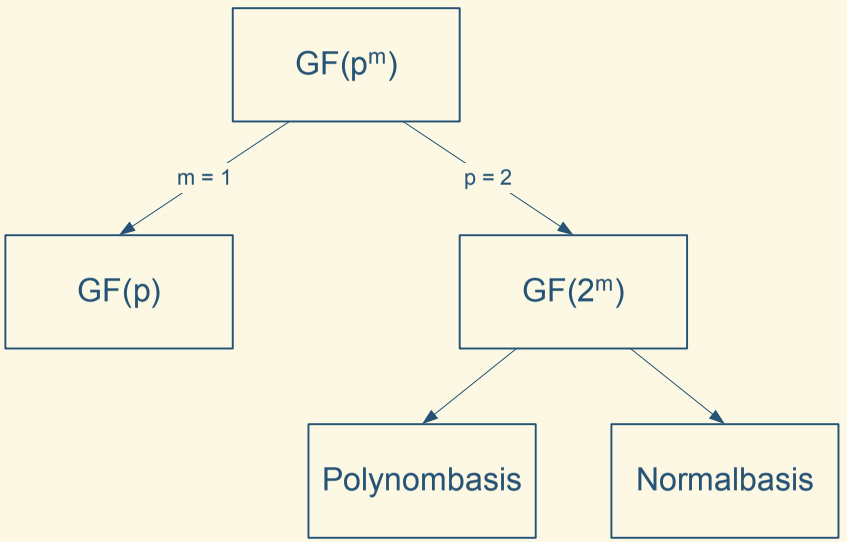
- Elliptische Kurven werden auf endliche Körper GF(q) abgebildet

- Elliptische Kurven dürfen nicht singulär sein ($\Delta \neq 0$)

Endliche Körper GF(q)

- Für endliche Körper sind zwei Bezeichnungen bekannt ($GF(q) = \mathbb{F}_q$)
- Wobei $q = p^m$, p ist eine Primzahl
- In der Kryptographie finden die beiden Sonderfälle m = 1 oder p = 2 Anwendung
- Für Hardwarelösungen ist der Sonderfall p = 2 interessant
- $char(GF(2^m)) = 2$

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



Punktoperationen

- Punktaddition $P \neq Q$
- Linie durch P und Q

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

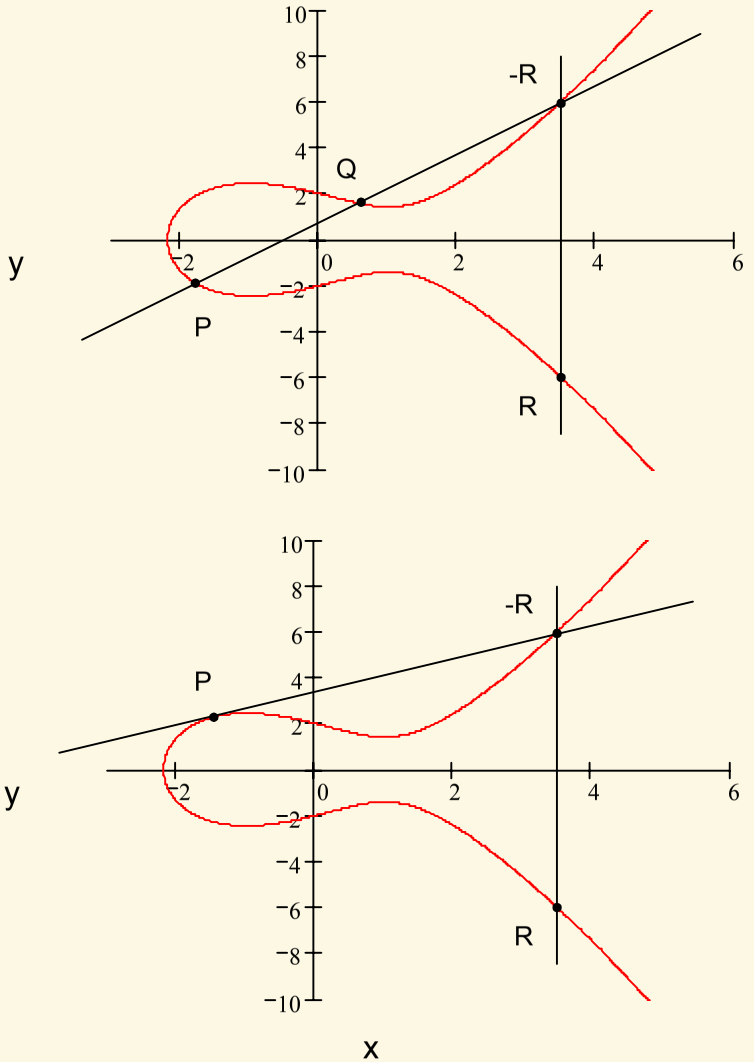
- Punktverdopplung $P = Q$
- Tangente an P

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

- Berechnung von R

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

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$



Algorithmus für die Skalarmultiplikation k*P

- Einfachster Algorithmus (Double and Add)

Input: Integer $k > 0$ und Punkt P

Output: $Q = k * P$

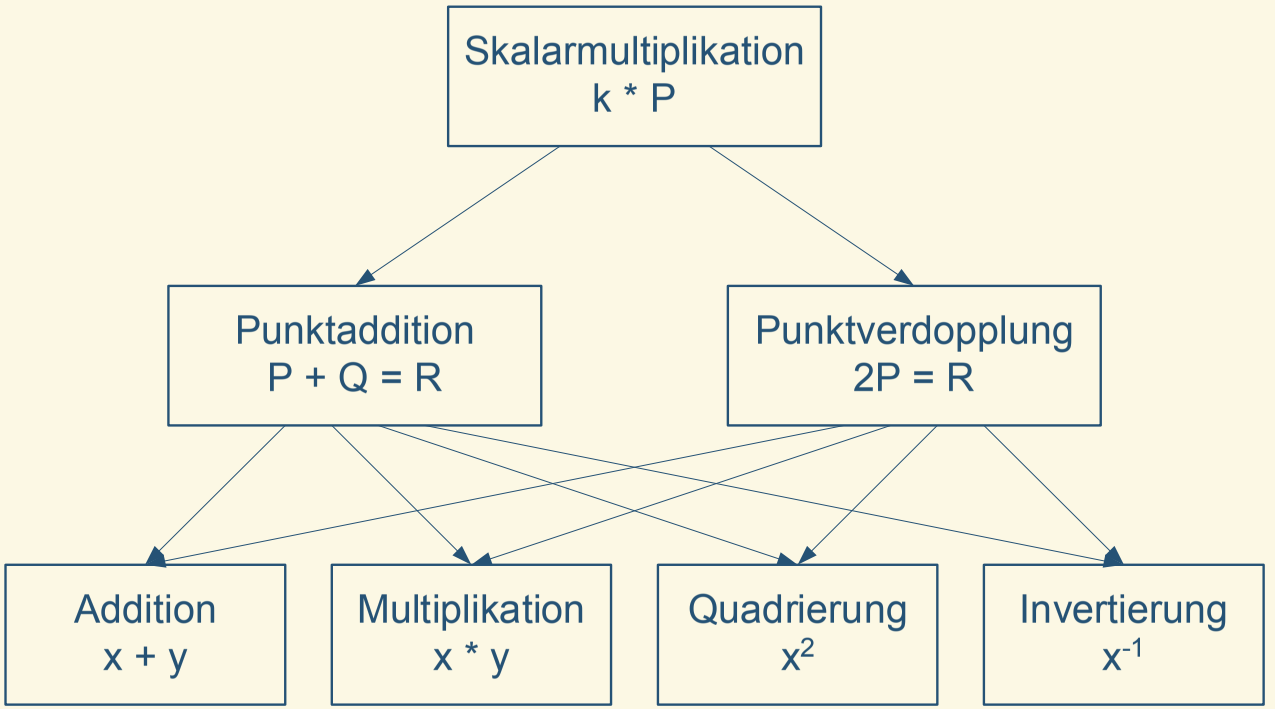
1. $k = (k_{n-1}, ..., k_1, k_0)_2$
2. $Q = P$
3. For i in (n - 2) downto 0 do
4. $\quad Q = 2 * Q$
5. $\quad$ If k_i = "1" then
6. $\quad\quad Q = Q + P$
7. EndFor
8. Return Q

Operationen im endlichem Körper GF(2^m)

- Benötigte Operationen im endlichem Körper GF(2^m)
  - Addition, Multiplikation, Division
- Quadrierung ist eine Sonderform der Multiplikation
  - Aber schneller berechenbar als Multiplikation
- Division ist sehr aufwendig zu berechnen
- daher Berechnung des multiplikativen Inversen, denn

  $a / b = a * b^{-1}$
- Endgültige Operationen im endlichem Körper GF(2^m)
  - Addition, Multiplikation, Quadrierung, Invertierung

Skalarmultiplikation k * P



Skalarmultiplikation

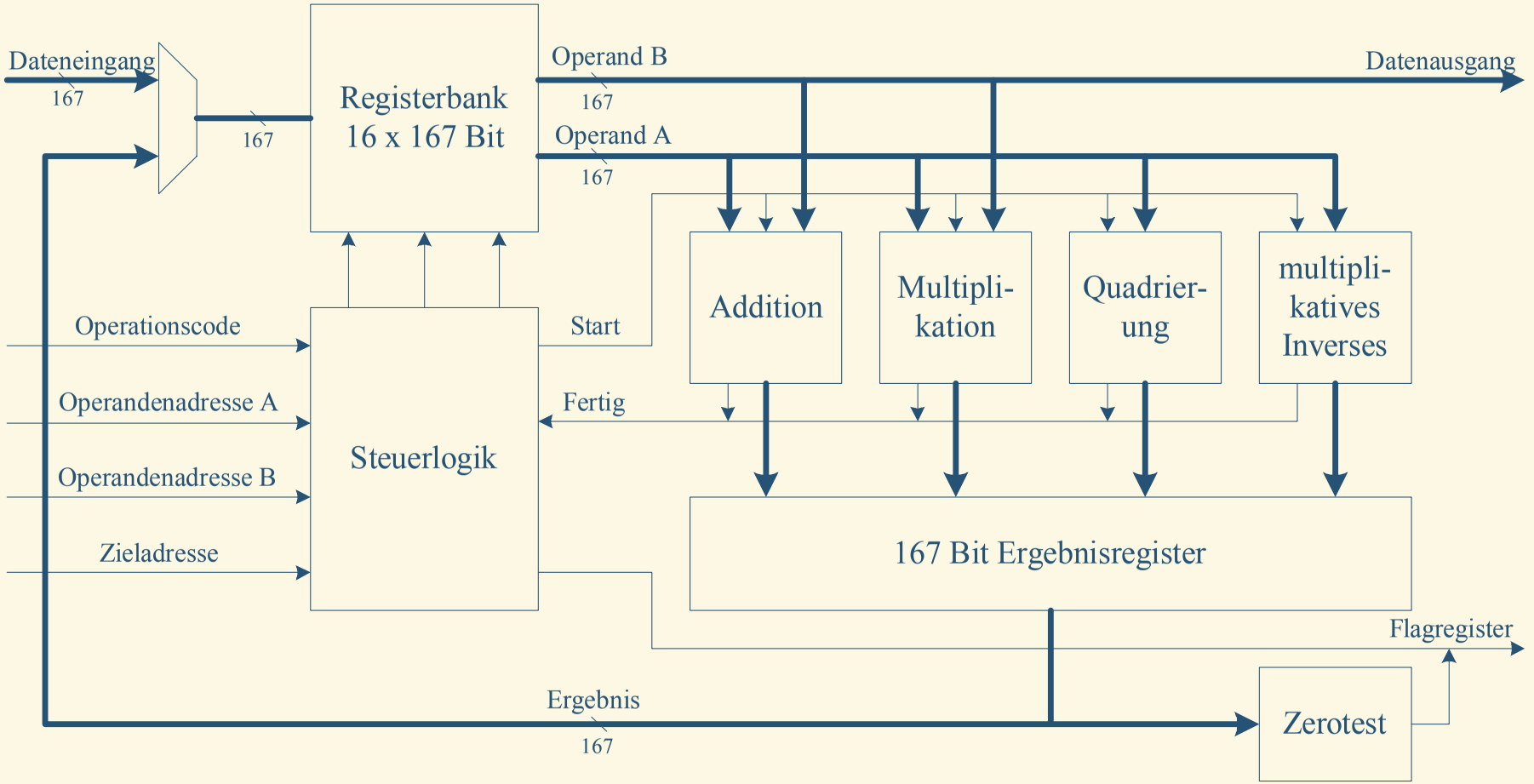
Punktoperationen auf der elliptischen Kurve

Operationen im endlichem Körper

Aufbau der ALU

- ALU für endlichen Körper GF(2^m) auf Polynombasis
  - irreduzibles Polynom: $x^{167} + x^6 + 1$
- Vier Operationen im endlichem Körper
  - Addition, Multiplikation, Quadrierung, Invertierung
- Speicher für Anfangswerte, Zwischen- und Endergebnisse
  - Les- und schreibbar
- Steuerlogik, steuerbar über Operationscode

Aufbau der ALU



Addition

- Addition im endlichem Körper $GF(2^m)$
- Grundlegendste Funktion
- Es gibt nur die Elemente 0 und 1

$$a + a = 2a \bmod 2 = 0a = 0$$

- Addition der einzelnen Bitstellen ohne Überlauf
- XOR-Verknüpfung der beiden Zahlen

- In einem Takt berechenbar

Multiplikation

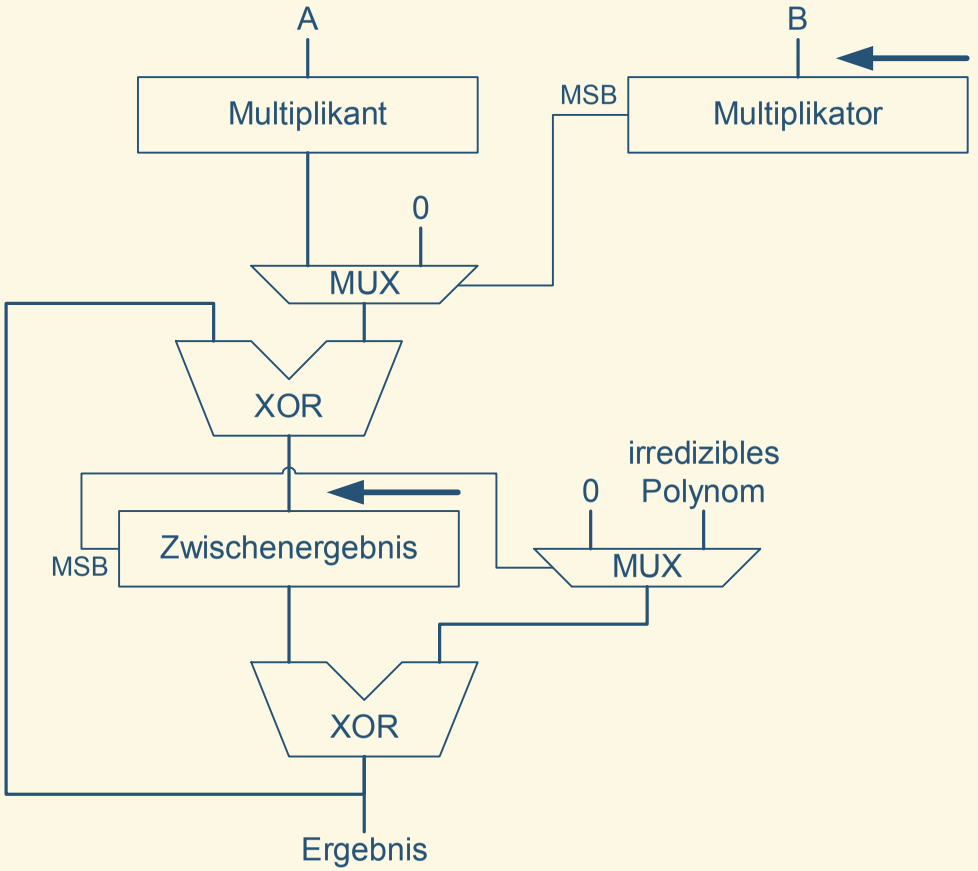
- Multiplikation von A und B, jeweils m Bit breit
- Ergebnis wird (2m – 1) Bit breit
- Reduktion mit dem irreduziblen Polynom auf m Bit

- Serielle Multiplikation wie Schulmethode
- Aufsummieren der Partialprodukte

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

Multiplikation

- Bekannte Architektur
  - Aber Addition = XOR
- Einbeziehung der Reduktion bei der Aufsummierung
- Kein Nachträgliche Reduktion nötig
- Benötigt m Takte

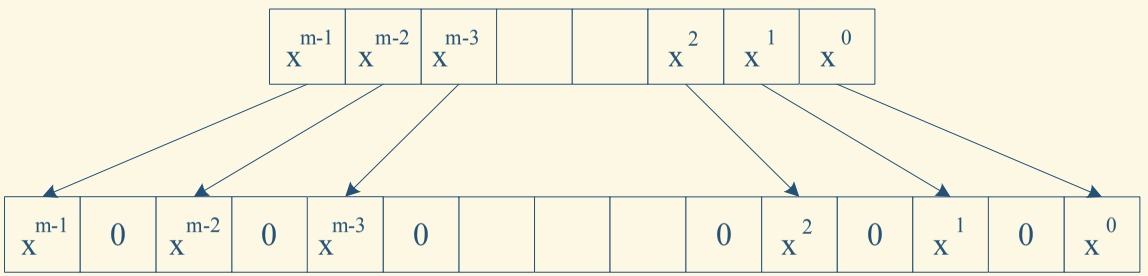


Quadrierung

- Quadrierung stellt Besonderheit dar, denn es gilt

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

- Ergebnis sehr schnell berechenbar, aber (2m - 1) Bit breit
- Anschließende Reduktion notwendig



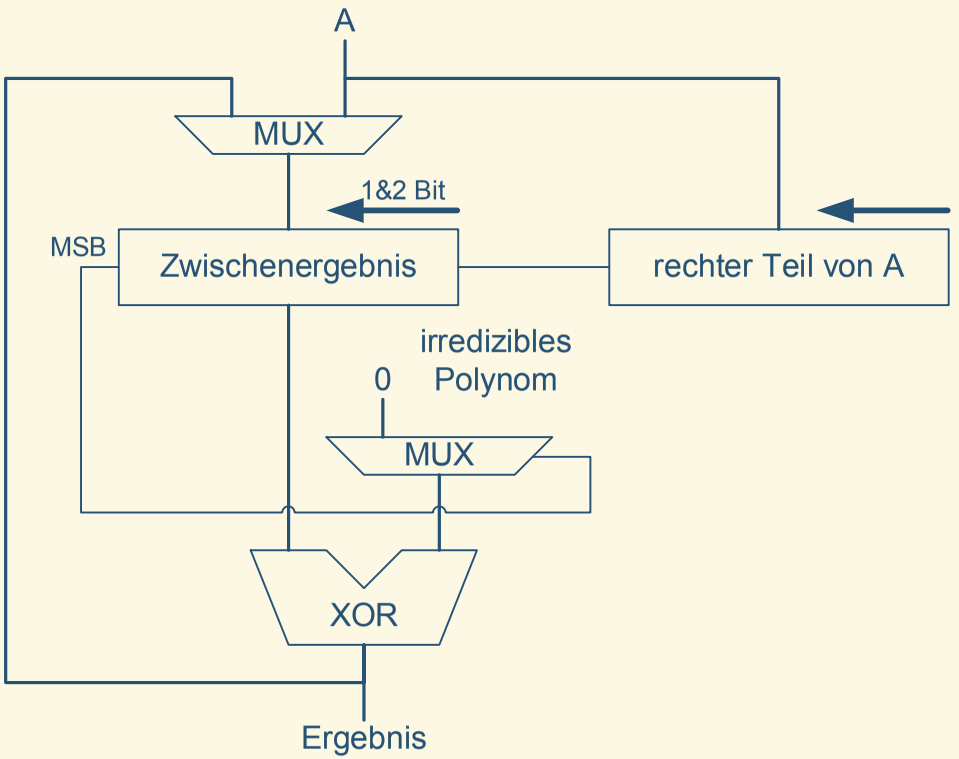
Quadrierung

- Reduktion wird seriell durchgeführt

- Ähnlichkeit mit Multiplikation
- Aber fast doppelt so schnell

- Benötigt (m + k – 1)/2 Takte
  - Irreduzibles Polynom: $x^m + x^k + 1$
  - Für unsere ALU

$$(167 + 6 - 1)/2 = 86$$



Invertierung

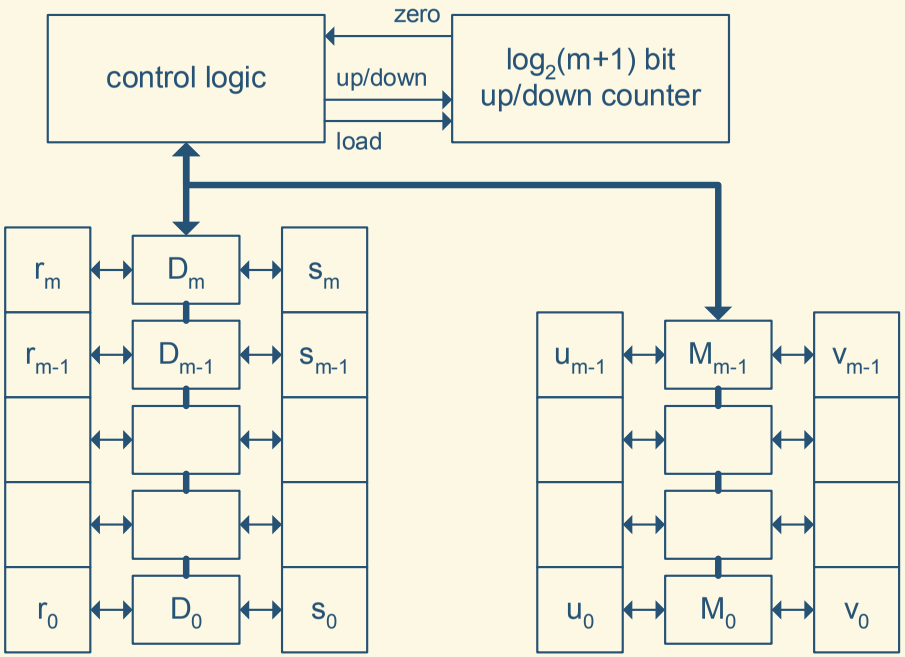
- Es gibt zwei Verfahren zur Berechnung des multiplikativen Inversen
  - Erweiterter Euklidischer Algorithmus
  - Fermat‘sches Theorem

- Es gilt $a \cdot a^{-1} = 1$
- Daher muss Eingang ≠ 0 sein

- Erweiterter Euklidischer Algorithmus eignet sich für Hardwarelösung am besten

Invertierung

- Verfahren von Brunner, Curiger und Hofstetter
- Geeignet für endliche Körper $GF(2^m)$ – Polynombasis
- Kleinste dokumentierte Hardwarerealisierungen
- Berechnung in m Takten
- Leicht parallelisierbar



Ergebnisse und Ausblick

- Synthese auf Xilinx FPGA XC400E-8
  - Taktfrequenz: 51,3 MHz
  - FlipFlops: 1419
  - Gatteräquivalente: 63049

- Erweiterung der ALU zu einem Prozessor für Kryptographie auf Basis elliptischer Kurven im Rahmen einer Diplomarbeit
  - Mikrocode gesteuerte Punktoperationen
  - Skalarmultiplikation
  - Interface zum LEON Prozessor

Zusammenfassung

- Vorstellung der Kryptographie auf Basis elliptischer Kurven
- Anforderungen an die ALU und ihr Aufbau
- Erste ALU mit integrierter Invertierungsfunktion
- Dadurch bis zu 3 mal schnellere Algorithmen möglich
- Arbeit an der Erweiterung zu einen kompletten Prozessor für Kryptographie auf Basis elliptischer Kurven