Großer Beleg

Arithmetische Optimierung eines kryptographischen Algorithmus für moderne FPGA-Architekturen

Peter Heinzig

17. Januar 2011

Überblick

Einleitung

Warum Kryptographie?

- Schutz vor Wirtschaftsspionage
- Schutz vor Überwachung
- Authentizität von Nachrichten

Symmetrische

- Schlüssel zum ver- und entschlüsseln gleich
- Beispiele: Cäsar, One-Time-Pad, DES, 3DES, AES

Asymmetrische

- Auch: Public-Key-Verfahren
- Verschiedene Schlüssel zum ver- und entschlüsseln
- Beispiele: RSA, El-Gamal, Diffie-Hellman-(Merkle-)Schlüsselaustausch, Rabin-Kryptosystem

DES

- Data-Encryption-Standard, Wettbewerb NIST
- Rundenbasierende Chiffre
- Heute: Nutzung von 3DES und AES

Rundenfunktion

Linke und rechte Seite werden getrennt verarbeitet

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus \mathrm{P}(\mathrm{S}(\mathrm{E}(R_i) \oplus K_i)) \end{aligned}$$

Hardwareimplementierung

FFs, Verdratung, ROM/LUTs, XOR

RSA

- Rivest, Shamir und Adleman, 1977
- Sicherheit hängt von Komplexität der Faktorisierung ab

Parameter

- p, q Primzahlen, $n = p \cdot q$
- $d \in \mathbb{Z}_{\varphi(n)}$, $e = d^{-1} \mod \varphi(n)$
- Public Key: n, d
- Private Key: n, e

Verschlüsselung

$$\begin{aligned} c &= m^d \mod n \\ m &= c^e \mod n \\ &= m^{d \cdot e} \mod n = m \end{aligned}$$

El-Gamal

- Taher El-Gamal, 1985
- Sicherheit abhängig vom Diskreter-Logarithmus-Problem

Parameter

- p Primzahl, Erzeugendes Element der Gruppe $\mathbb{Z}_{p^*}$: g
- Zufallszahl a, $A = g^a \mod p$
- Public Key: p, g, A
- Private Key: p, a

Verschlüsselung

Weitere Zufallszahl: b

$$c = (B = g^b, m \cdot A^b) \mod p$$
$$m = m \cdot A^b \cdot B^{p-1-a} \mod p = m$$

Diffie-Hellman-Merkle-Schlüsselaustausch

- Diffie, Hellman, Merkle, 1976
- Verfahren zum Schlüsselaustausch
- Sicherheit abhängig vom Diskreter-Logarithmus-Problem

Parameter

- p Primzahl, Erzeugendes Element der Gruppe $\mathbb{Z}_{p^*}$: g
- Einigung auf p, g

Ablauf

Zufallszahlen: a, b, $A = g^a$, $B = g^b$, Übertragung von A und B

$$\begin{aligned} s &= A^b \mod p \\ &= B^a \mod p \\ &= g^{a \cdot b} \mod p \end{aligned}$$

Kernfunktion

PowerMod: $a^b \mod c$

Nutzbare arithmetische Funktionen auf FPGAs

- Addition, Subtraktion
- Shiften
- Multiplikation mit kleinen Faktoren

Wie kann man PowerMod auf diese Funktionen reduzieren?

- Aufsplitten von PowerMod
- Modulo ist der Rest einer Division und Ziffern iterativ berechenbar
- Exponentiation ist eine Verkettung von Multiplikationen
- Multiplikationen lassen sich als Vielfachaddition darstellen

Ansätze zur Reduktion von PowerMod

- Naiv: Zuerst Exponentiation, dann Modulo
  $\rightarrow$ Zwischenergebnis wird sehr groß
  $\rightarrow$ Berechnung ist langsam, Hardwarebedarf hoch
- Verschiebung der Modulorechnung in die Multiplikationen
  $\rightarrow$ Operanden bleiben klein
  $\rightarrow$ schneller, weniger Hardware
  Aber: MultiplyMod wird als Funktion benötigt

Aufsplitten der Exponentiation

- Naiv: Mehrfache Multiplikationen der Basis mit sich selbst
  $\rightarrow$ viele Berechnungsschritte (linear) $\rightarrow$ langsam
- Square-and-Multiply: Berechnung von Quadraten und Multiplikation dieser
  $\rightarrow$ weniger Berechnungsschritte (logarithmisch) $\rightarrow$ schnell

Arten von Square-and-Multiply

LSB-first

- Schrittweise Berechnung über $a^b \mod 2^n$, a^{2^n}
- Parallelisierbar (benötigt doppelte Hardware)

MSB-first

- Berechnung über $a^{\lfloor \frac{b}{2^n} \rfloor}$
- Nicht parallelisierbar

Berechnung von MulitplyMod

Ansätze

- Naiv: Erst Multiplizieren, dann Modulo
  $\rightarrow$ Zwischenergebnis doppelt so groß
- Iteratives Einbinden von Modulo
  - Montgomery-Algorithmus
  - Eigene Implementierung: MSB-first-„Double-and-Add“

Montgomery-Algorithmus

- Berechnet iterativ $a \cdot (b \mod 2^n) \cdot 2^{-n} \mod m$
- Wahl der Korrekturgröße abhängig vom LSB
- Korrektur des Ergebnisses erforderlich

Eigene Implementierung

- Berechnet iterativ $a \cdot \left\lfloor \frac{b}{2^n} \right\rfloor \mod m$
- Wahl der Korrekturgröße abhängig vom MSB
- Keine Korrektur nötig

Berechnungsschritt der Implementierung

Vorschrift

$$\begin{aligned} a \cdot \left\lfloor \frac{b}{2^{n+1}} \right\rfloor &= 2a \cdot \left\lfloor \frac{b}{2^n} \right\rfloor + b_{n+1} \cdot a \mod m \\ &= 2a \cdot \left\lfloor \frac{b}{2^n} \right\rfloor + b_{n+1} \cdot a - k \cdot m \end{aligned}$$

Schwierigkeiten

- Ermittelung des k unter Berücksichtigung möglichst weniger Bitstellen
- schnelle Addition/Subtraktion bei üblichen Bitbreiten von 1024 Bit

Anpassungen

- Aufweichen der strengen Vorschrift für den Operanden
  $\rightarrow$ k darf auch um 1 zu klein sein
  $\rightarrow$ Korrektur des Ergebnisses evtl. nötig
- Shiften von Modul und 1 Operand
- Addition/Subtraktion zu einer Reduktion umbauen
  $\rightarrow$ Vermeiden des langen Carry-Pfades
  $\rightarrow$ Abschließende (iterative) Aufsummierung nötig
- Abschließende Berechnungen lassen sich verbinden

Parameter

- Operandenbreite
- Anzahl der Binärstellen pro Takt (Multiplikation)
- Anzahl der Binärstellen pro Takt (Addition)

Implementierung

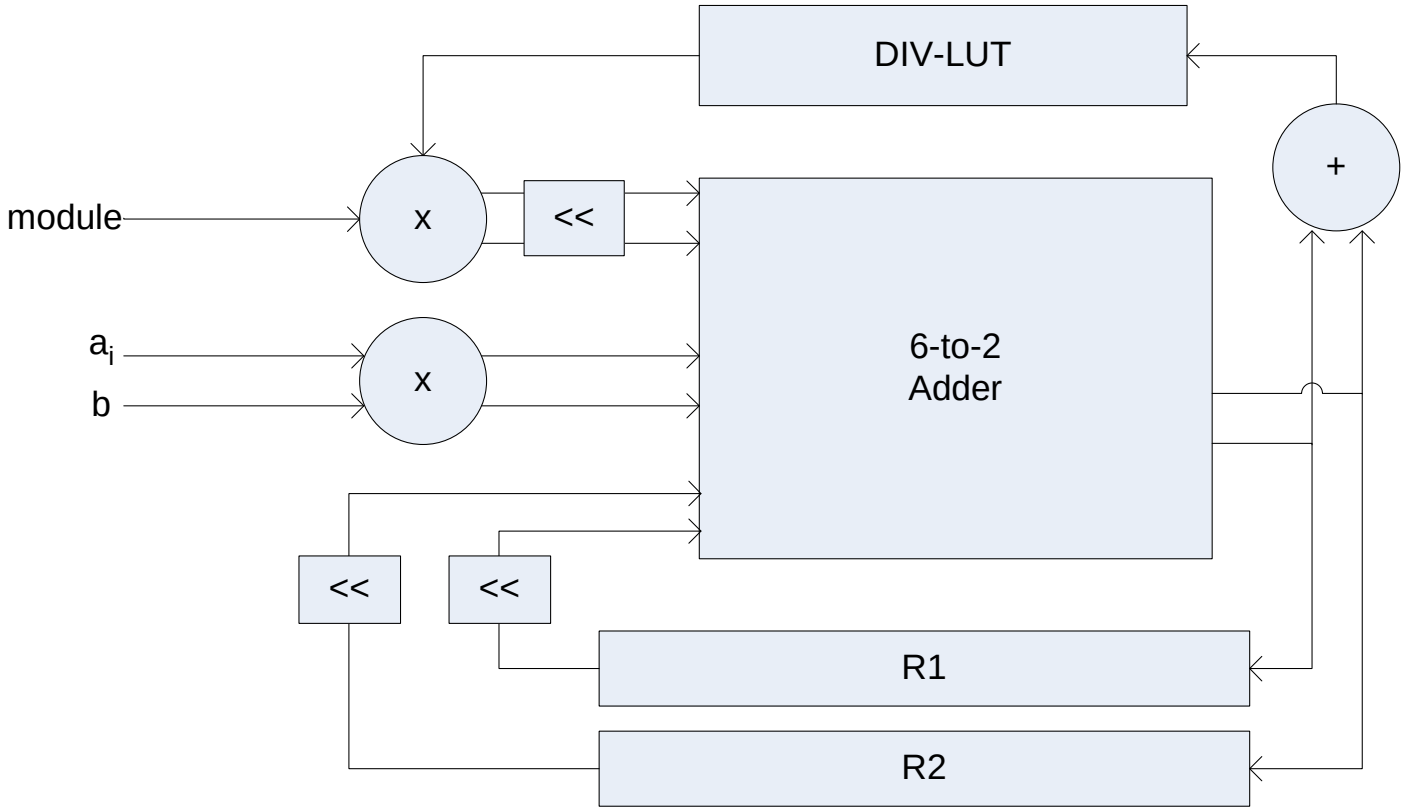

Abbildung: Multiply-Mod-Design mit Carry-Save-Implementierung

Weitere Module

log4prefix

Ein Modul zur Bestimmung der höchstwertigen 1

FastShift

Ein schneller Shifter, der eine freie Verschiebung von Zwischenergebnissen in 1 Takt ermöglicht

reduce6to2

Ein Reduzierer, der 6 Additionsoperanden nimmt und diese zu 2 Operanden verknüpft. Dabei ist keine Carry-Weiterleitung nötig.

Syntheseergebnisse

Auf einem Virtex 5 VFX200T:

Hardwarebedarf

clk	145,4 MHz
SliceReg	15465
SliceLUT	71282

Berechnungsdauer für 1024 Bit

Takte

LUT-Berechnung für Division	128
Shiften des Moduls	1
Multiplikation Multiply-Mod	206
Addition Multiply-Mod	13
Multiply-Mod (Summe)	219
Power-Mod	225409

Zeitbedarf

- Möglicher Takt: 145,4 MHz
- Zeitbedarf für 1 Operation mit 1024 Bit-Operanden: $\frac{225409}{145{,}4\text{MHz}} = 1,55$ ms

Vergleich

Design	Device	Resourcen	ms
eigenes	Virtex5	15465 Reg, 71282 LUT	1,55
Ref1	Virtex2	14334 Slices, 62 18-Mul	2,33
PC	Phenom II X4	3,2GHz	3,00
Ref2	XC40250XV	6633 CLBs	11,95
Ref3	Stratix EP1S40	434LE, 13,6k Bits, 4 DSP	26,39
Ref4	Virtex-E 2000-8	1188 Slices	208,00

Vor- und Nachteile

Vorteile

- Sehr schnell
- Parametrierbar
- Auch gerade Module nutzbar

Nachteile

- hoher Hardwareaufwand

Danke

vielen Dank für die Aufmerksamkeit