TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Technische Informatik, Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

Untersuchung zur Realisierung des Hashalgorithmus Scrypt auf einer energieeffizenten parallelen Plattform

Vortrag zur Belegverteidigung

Franz Gregor

franz.gregor@mailbox.tu-dresden.de

Dresden, 13.05.2015

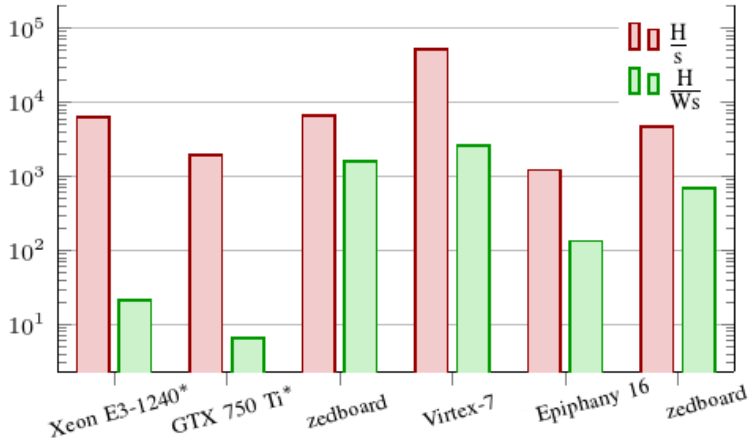
01 Einleitung
Passwort Hashing

$$h = H(p)$$
$$h \stackrel{?}{=} H(p')$$

Passwörter abzählbar → Brute-Force

- *H()* bestimmt Brute-Force Aufwand
- Kompromiss zw. Nutzerwartezeit & Angriffsdauer → ca. 100ms
- Problem: Moores Law
  - DEScrypt, md5crypt Heute nicht mehr sicher

01 Einleitung
Bcrypt



Bcrypt Performancevergleich [1]

- Veränderbarer Kostenparameter
  - an Moores Law anpassbar
- Problem: "Waffenungleichheit"
  - Betreiber Softwareimplementierung
  - Angreifer auch FPGA, GPU

01 Einleitung
Scrypt

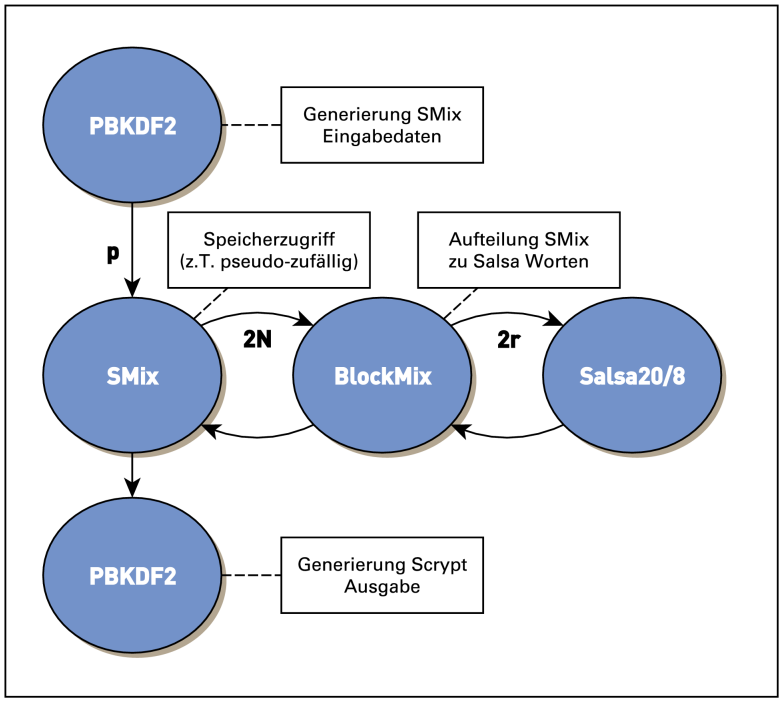
- veröffentlicht 2009 von Percival [2]
- Custom Hardware schwächen → viel Speicher

	Software	Custom Hardware
Dichte	hoch	niedrig
Latenz	hoch	niedrig
Bandbreite	niedrig	hoch
Architektur	getrennt (extern)	**integriert**

- Scrypt Kostenparameter bestimmt Aufwand UND Speichermenge

Gliederung

02 Analyse Scrypt
Funktionsweise

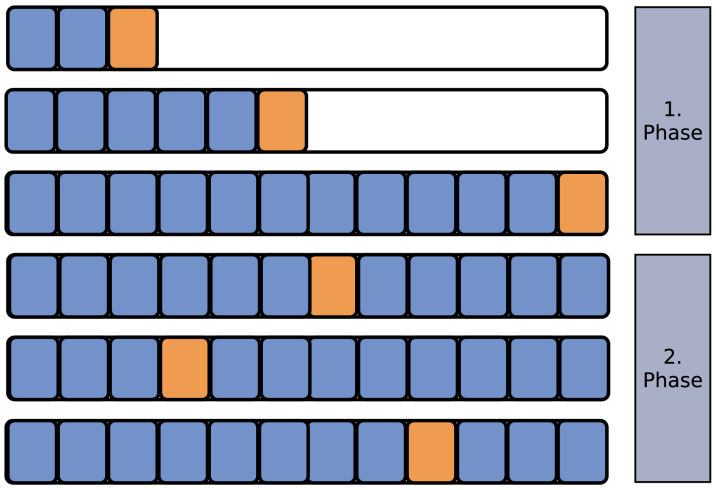


- 3 Parameter N, r, p
  - p... Parallelität (i.d.R. 1)
  - N... Speicherworte, Berechnungen
  - r... SMix-Wortbreite (2r*512 Bit)

02 Analyse Scrypt

SMix

```
function SMix(X):
  V...   array(0 to N)
  j...   int
  X,T... SMix-Wort
  for i in 0 to N do:
    V[i] = X;
    X = BlockMix(X);
  for i in 0 to N do:
    j = Integerify(X);
    T = X xor V[j];
    X = BlockMix(T);
  return X;
```



Speicherzugriffe SMix

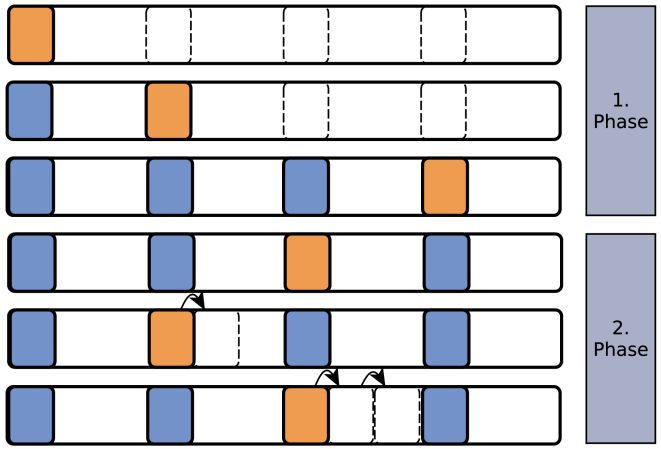
02 Analyse Scrypt
Speicherbedarf

- Scrypt Paper [2]: $N = 2^{14}$ o. $N = 2^{16}$, $r = 8$
  - 16MB o. 64MB
- Android: $N = 2^{15}$, $r = 3$
  - 12MB
- Litecoin: $N = 2^{10}$, $r = 1$
  - 128kB

- KC705 Evaluationsboard hat rund 2MB Block RAM!

02 Analyse Scrypt
Time-Memory Trade-Off (TMTO) nach [3]

```
function SMix(X):
  for i in 0 to N do:
    if i mod x == 0
      V[i] = X;
    X = BlockMix(X);
  for i in 0 to N do:
    j = Integerify(X);
    l = j - (j mod x);
    y = V[l];
    for m in l+1 to j do:
      y = BlockMix(y);
    T = X xor y;
    X = BlockMix(T);
  return X;
```

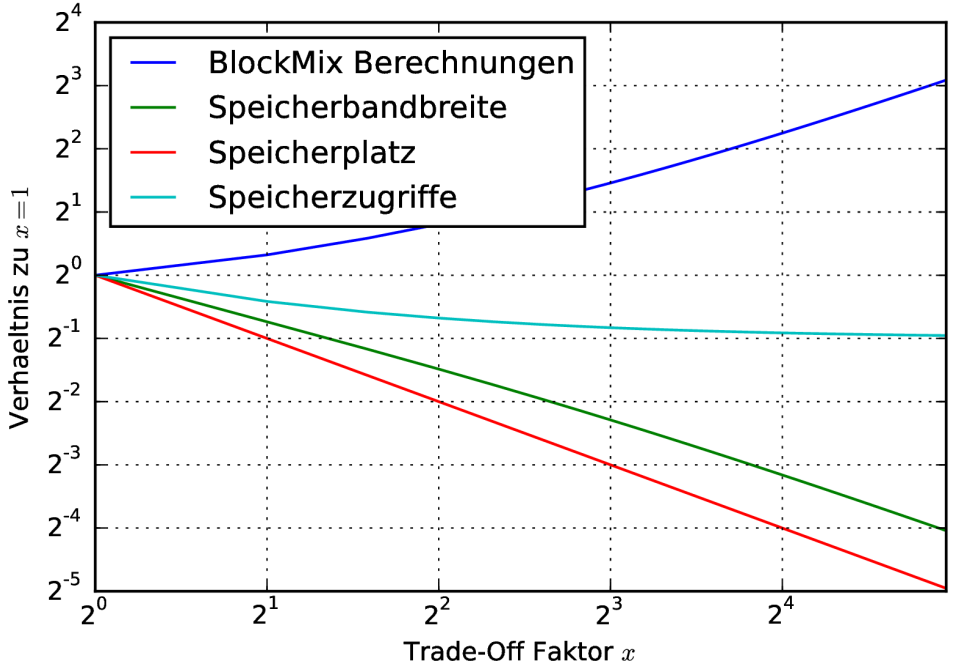


Speicherzugriffe SMix mit TMTO

- 1/x des Speichers
- N(x-1)/2 zusätzl. Berechnungen

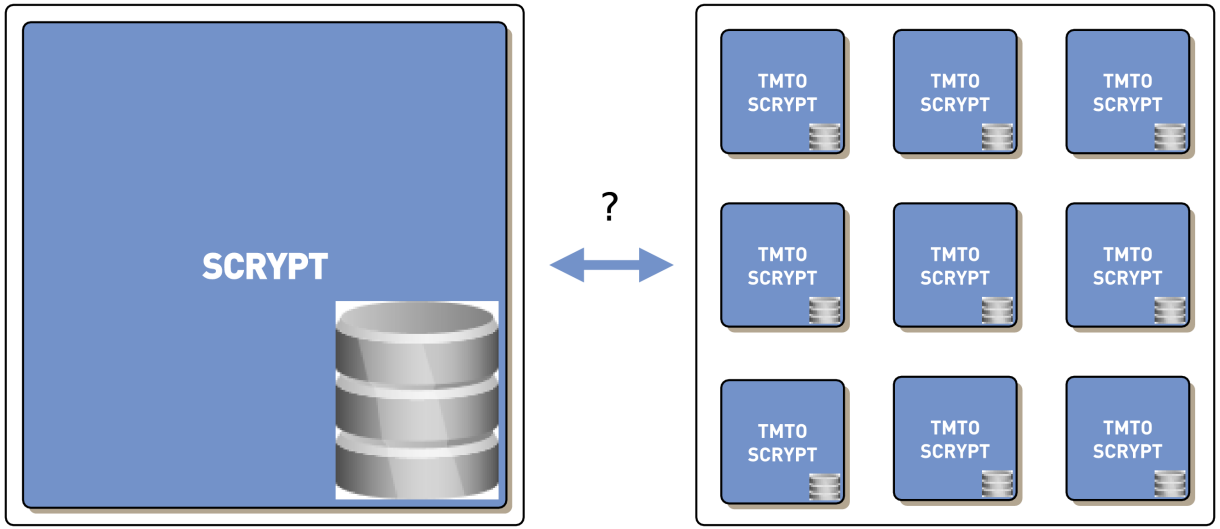
02 Analyse Scrypt
Time-Memory Trade-Off (TMTO)

- BlockMix() dominierende Berechnung
  - SMix Speicherbandbreite ≈ #Zugriffe/#BlockMix()
- Verhalten in Abhängigkeit von Trade-Off Faktor x:

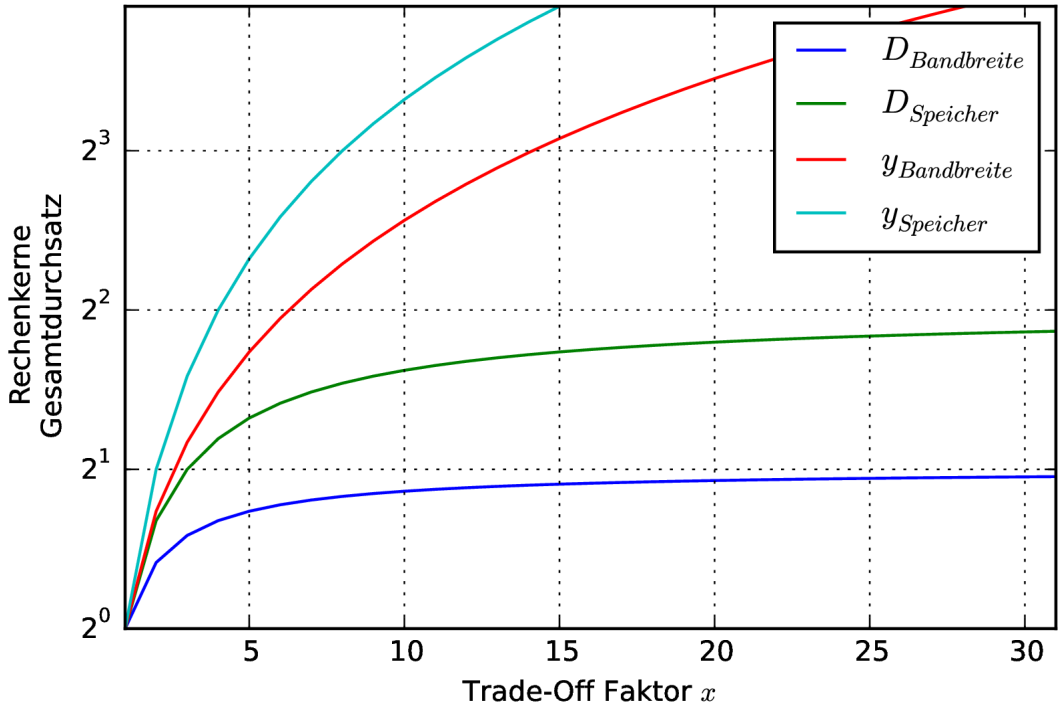


02 Analyse Scrypt
Time-Memory Trade-Off (TMTO)

- Trade-Off
  - Gibt Ressourcen frei
  - Ermöglicht mehr parallele Rechenkerne



02 Analyse Scrypt
TMTO - Leistungssteigerung!



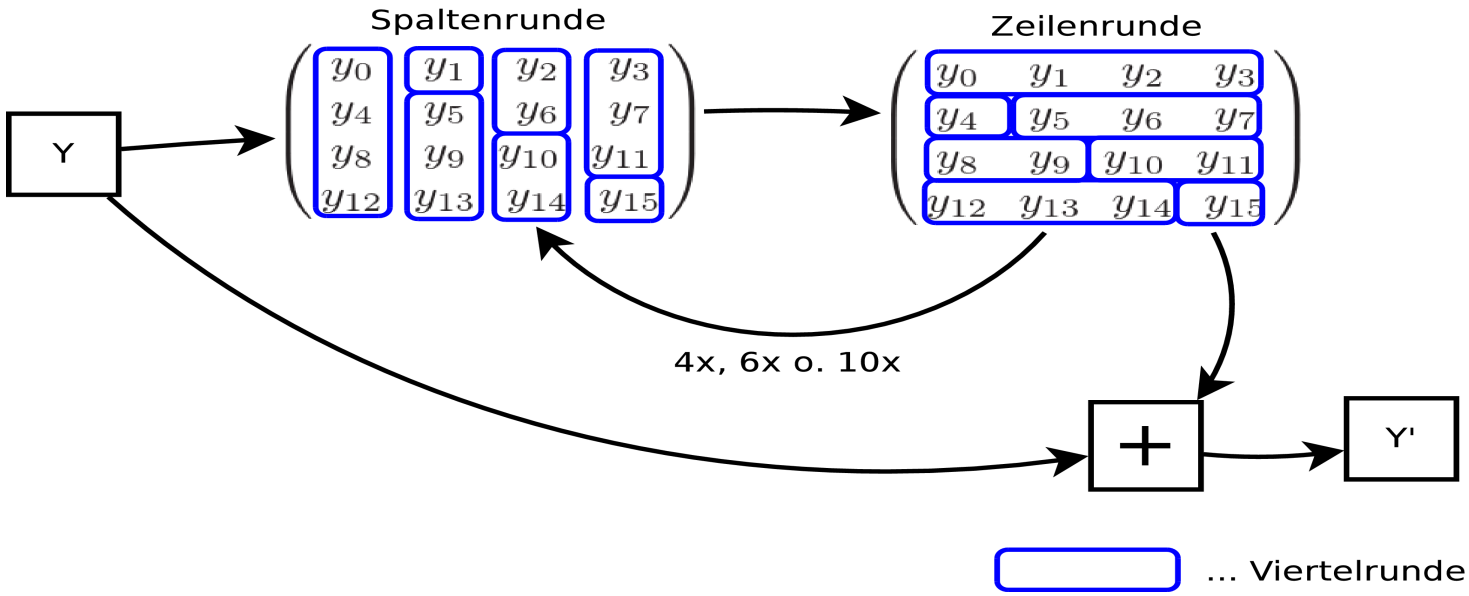
02 Analyse Scrypt
Zusammenfassung

- (Zu) hoher Speicherbedarf
- Lösbar mit TMTO
- Parallele Berechnung mit TMTO → höherer Gesamtdurchsatz
- Aber: BlockMix() bisher BlackBox

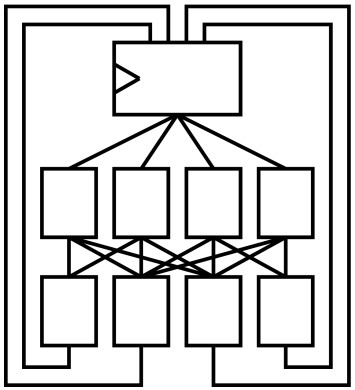
```
function BlockMix(B_0, B_1, … B_2r-1):
  X = B_2r-1;
  for i in 0 to 2r do:
    X  = Salsa20/8(X xor B_i);
    Y_i = X
  return (Y_0, Y_2, … Y_2r-2, Y_1, Y_3, … Y_2r-1);
```

03 Salsa
Funktionsweise

- Stromverschlüsselung, 2005 von Bernstein
- In Scrypt pseudo-zufällige Salsa Core Funktion

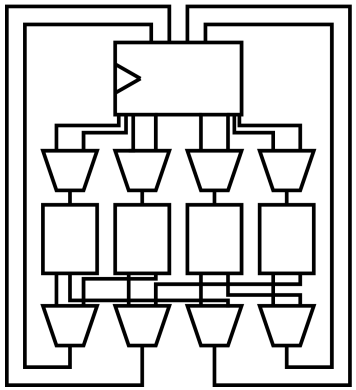


03 Salsa
Iterationsfunktionalität



Doppelrunde (8xQR)

- Langer kritischer Pfad
- Viele Ressourcen (LUTs)
- Geringe Taktanzahl (4)

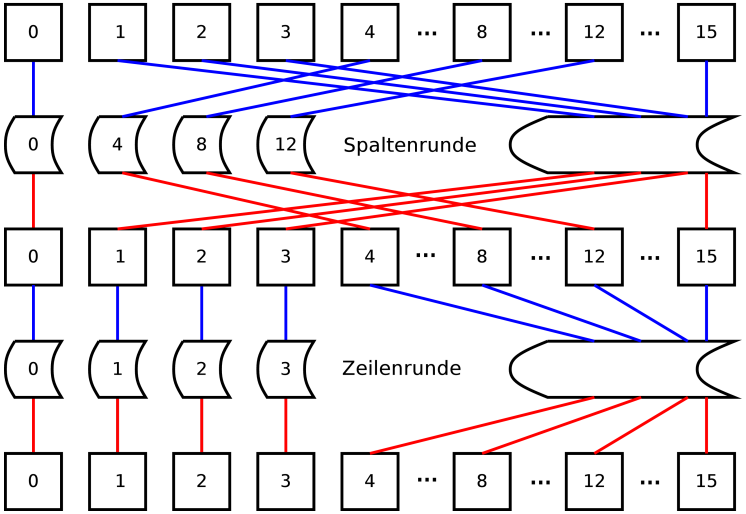


Generische Runde (4xQR)

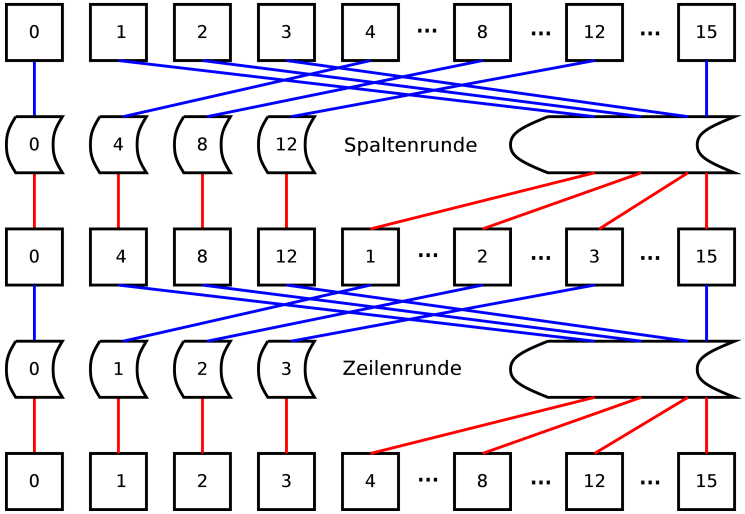
- Halber kritscher Pfad
- Halbe Ressourcen (LUTs)
- Doppelte Taktzahl (8)
- Leider nur Theorie:
  + 24 2-zu-1 Multiplexer

03 Salsa

Iterationsfunktionalität - Optimierung 4xQR



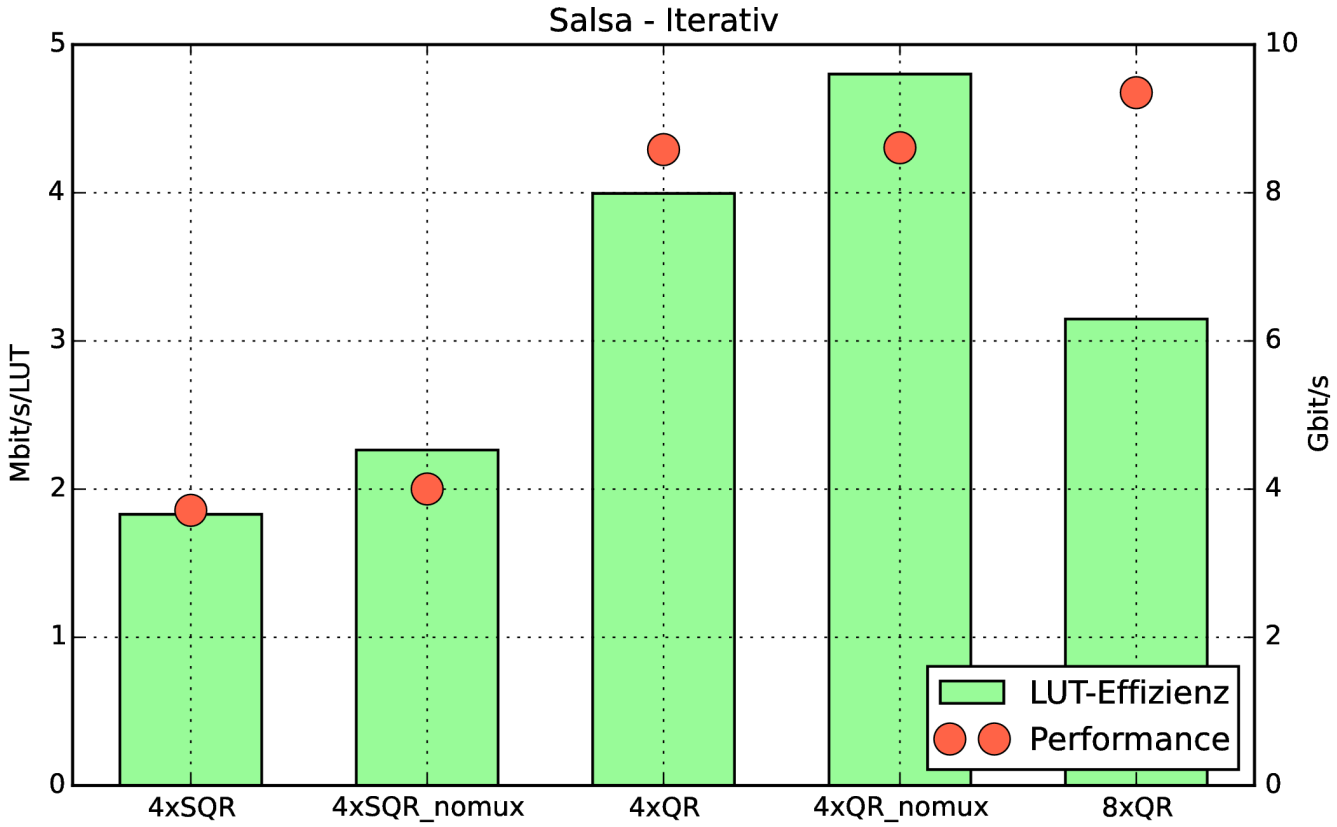
Ursprüngliches Auswahlnetzwerk



Optimierung (frei sortiertes Zwischenergbenis)

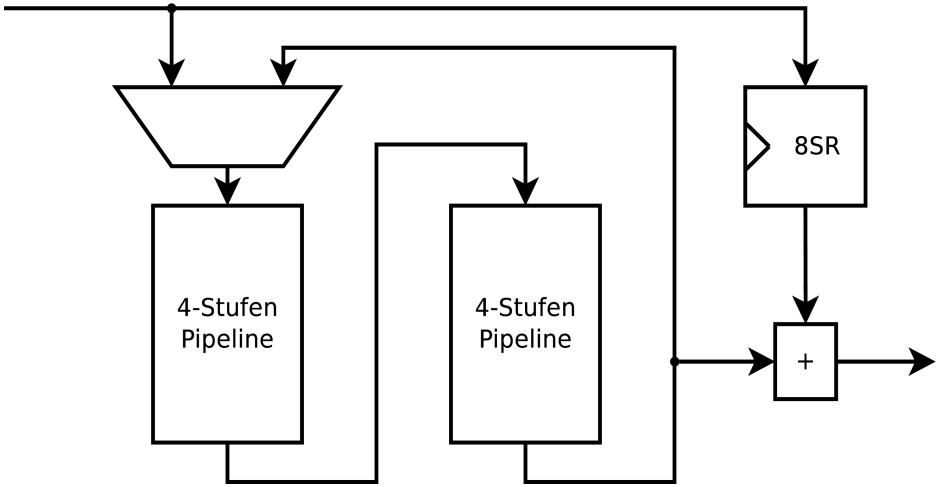
→ keine Multiplexer notwendig

03 Salsa
Ergebnisse

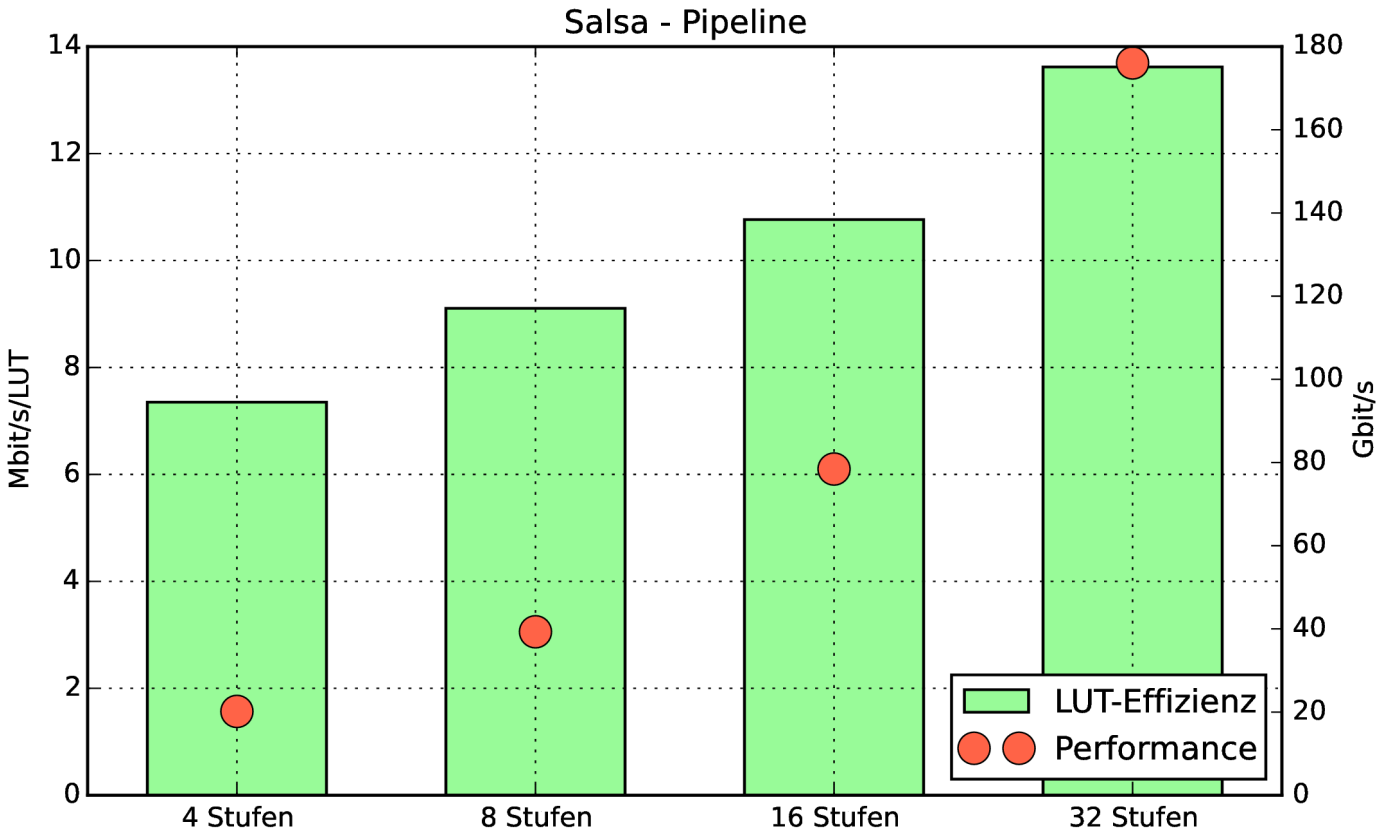


03 Salsa
Pipeline

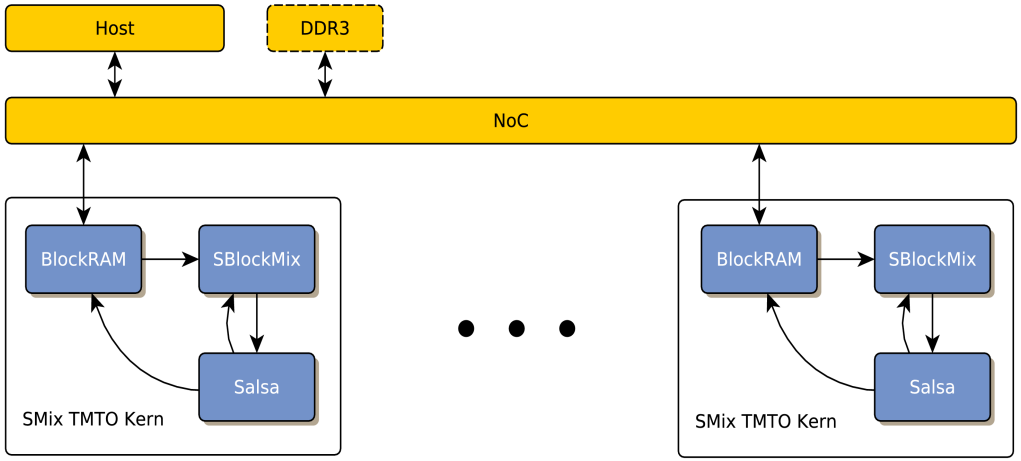
- Addition → Eingabedaten in Pipeline
- iterative, gepipelinte Architektur



03 Salsa
Ergebnisse

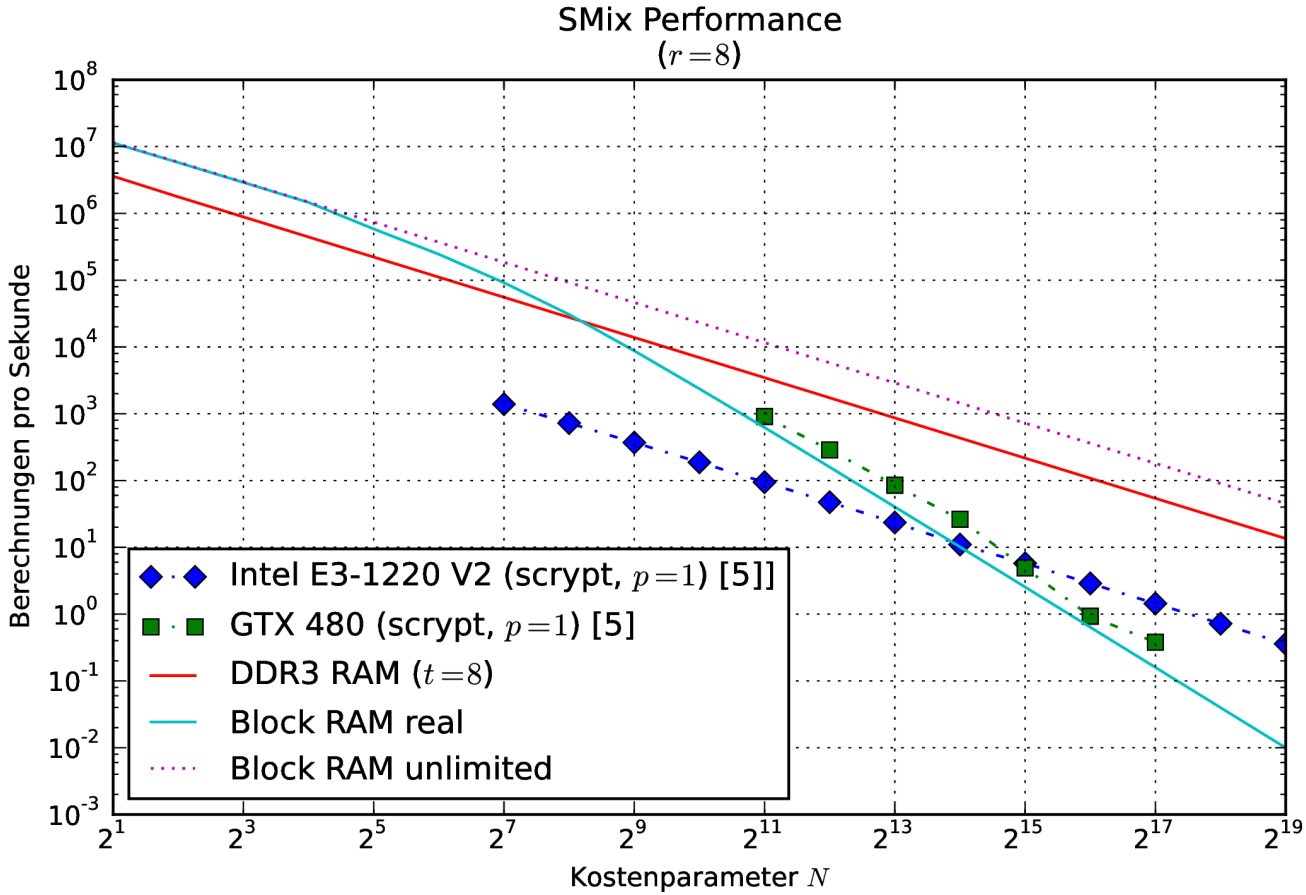


04 Entwurf & Abschätzung
Architektur



- Pro TMTO SMix Kern:
  - 8 x Block RAM (512 Bit Worte)
  - 4xQR_nomux (343 Slices auf Kintex 7)
- KC705 445 Block RAM → 55 SMix Kerne
- ML605 416 Block RAM → 52 SMix Kerne

04 Entwurf & Abschätzung
Vergleich



05 Zusammenfassung

- Scrypt
  - Ziel erreicht: Hardware schlechter als Software (bei großem Kostenparameter)
  - Aber:
    - Mit TMTO berechenbar
    - FPGA DDR3 System schneller als Software
- Salsa
  - Effizienz durch Optimierungen erhöht
  - Pipeline auf Xilinx FPGA effizient implementiert

06 Quellen

[1] Friedrich Wiemer, Ralf Zimmermann. High-Speed implementation of bcrypt password search using special-purpose hardware. In *2014 International Conference on Reconfigurable Computing and FPGAs (ReCoFig)*, 2014.

[2] Colin Percival. Stronger key derivation via sequential memory-hard functions. *Self-published,* 2009

[3] Christian Forler, Stefan Lucks, und Jakob Wenzel. Catena: A memory-consuming password scrambler. *IACR Cryptology ePrint Archive*, 2013:525, 2013.

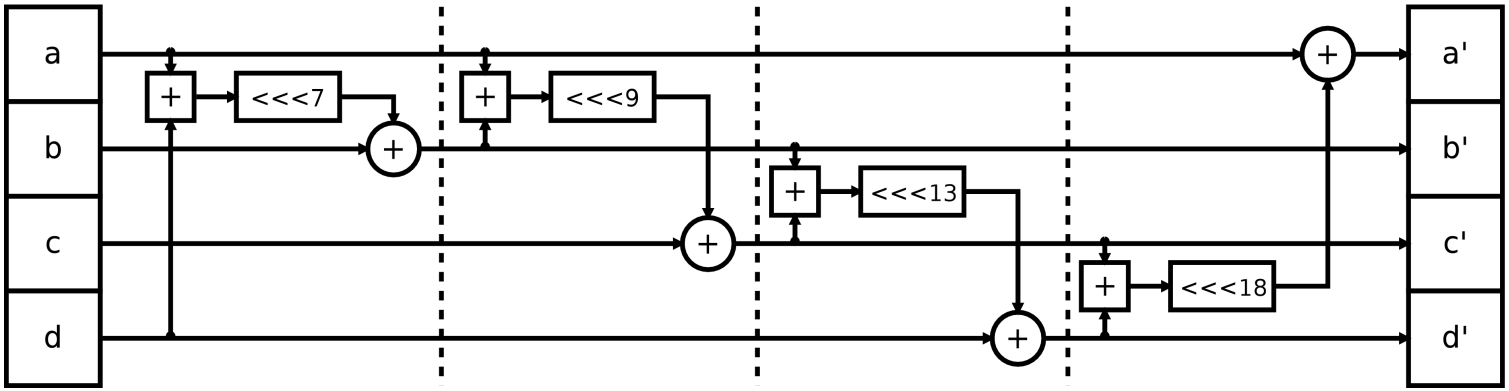
[4] Luca Henzen, Flavio Carbognani, Norbert Felber und Wolfgang Fichtner. *VLSI hardware evaluation of the stream ciphers salsa20 and chacha, and the compression function rumba. In Signals, Circuits and Systems, 2008. SCS 2008. 2nd International Conference on*, pages 1-5. IEEE, 2008.

[5] Markus Dürmuth, und Thorsten Kranz Thorsten. *On Password Guessing with GPUs and FPGAs,* 2014.

[6] Junjie Yan und Howard M Heys. Hardware implementation of the salsa20 and phelix ciphers. In *Electrical and Computer Engineering, 2007*. CCECE 2007.

03 Salsa

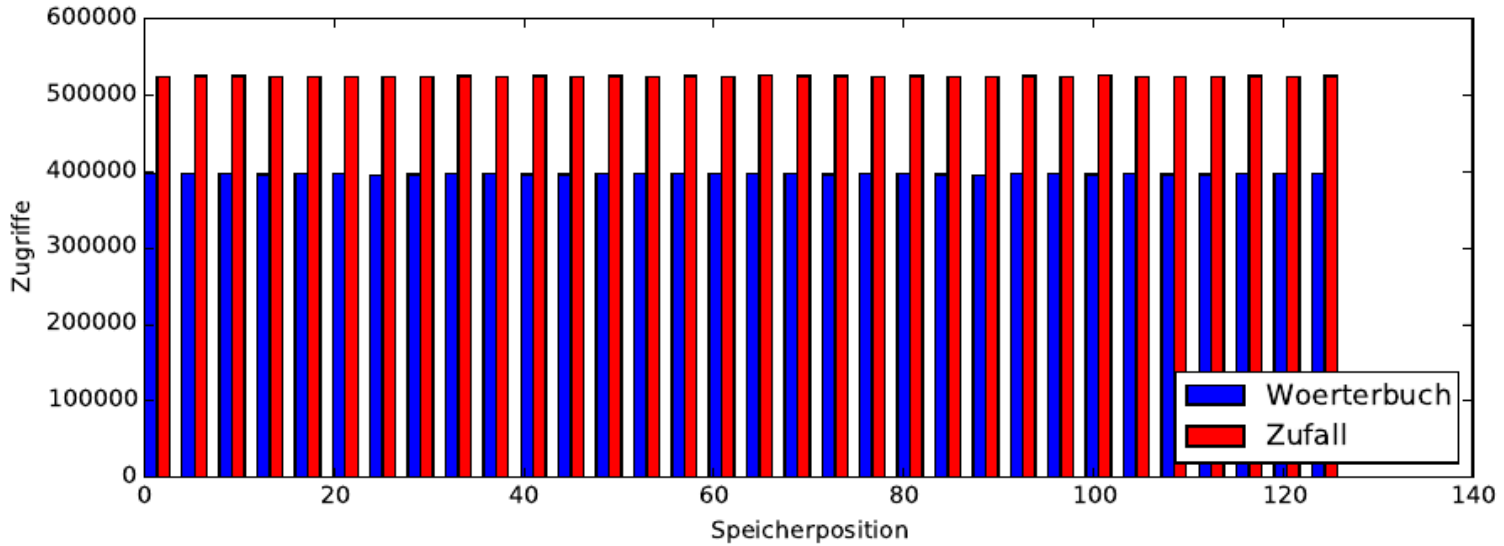
Iterationsfunktionalität (nach [4])



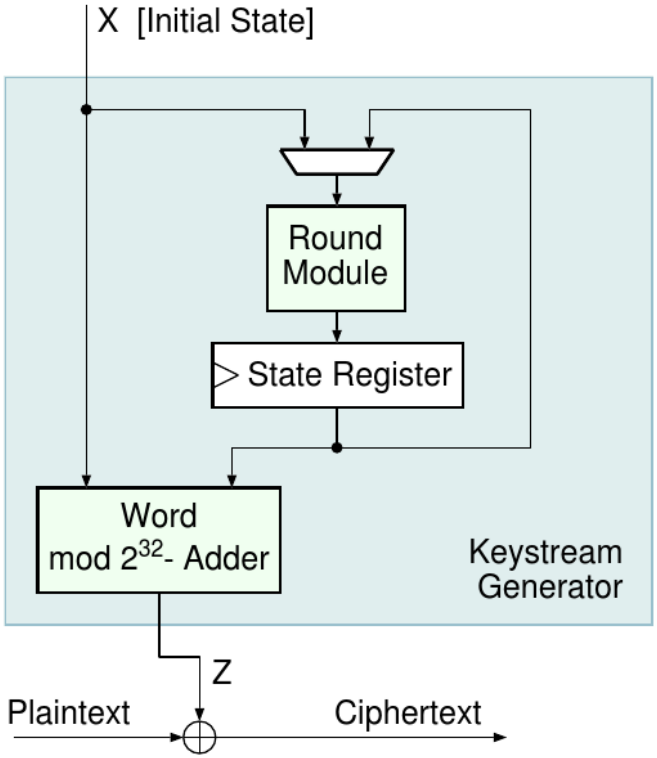
(4x)SQR

- ¼ kritischer Pfad
- ¼ Ressourcen (LUTs)
- höchste Taktanzahl (32)
- aber: zusätzl. Multiplexer

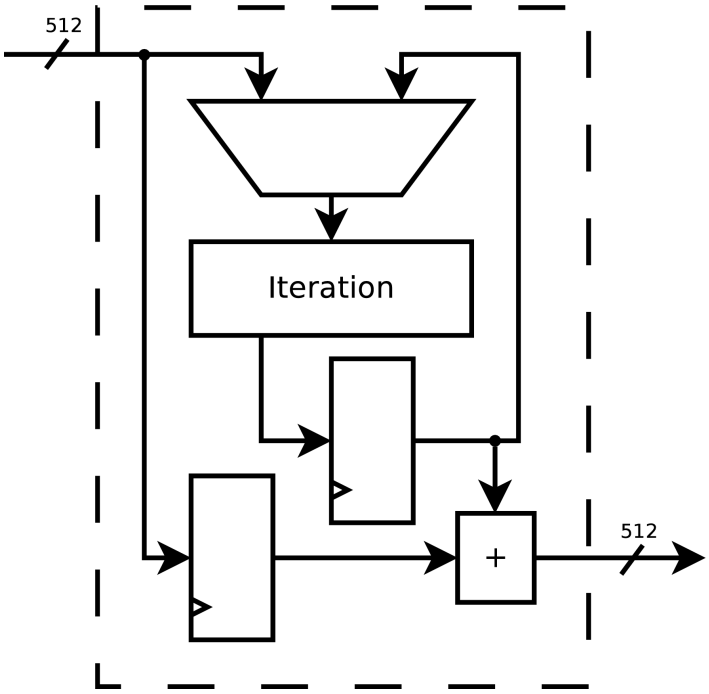
02 Analyse
Scrypt Caching



02 Salsa
Architektur



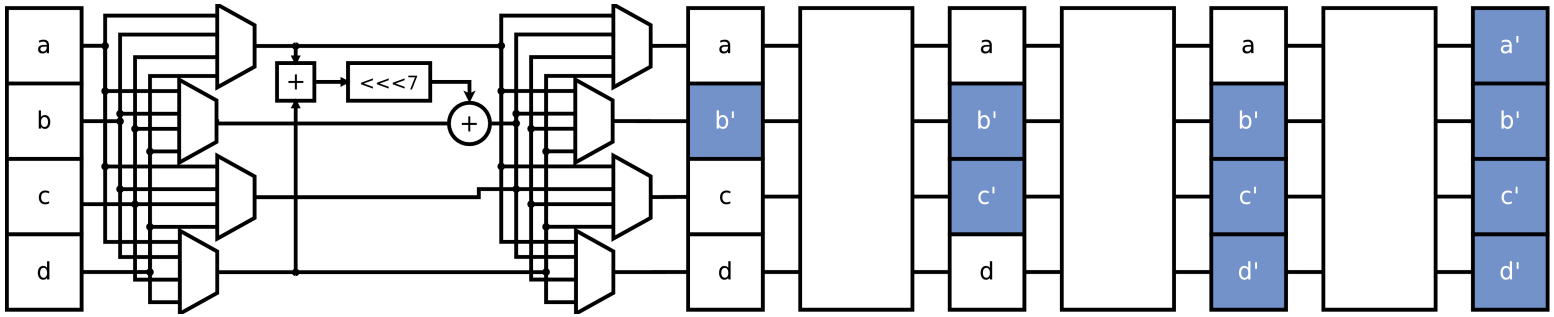
nach Henzen [4]



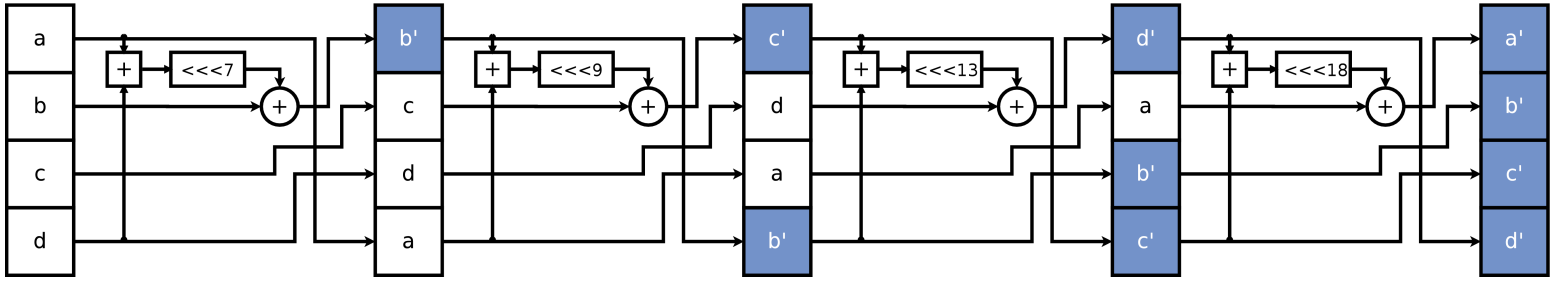
Eigene Architektur

03 Salsa

Iterationsfunktionalität - Optimierung 4xSQR



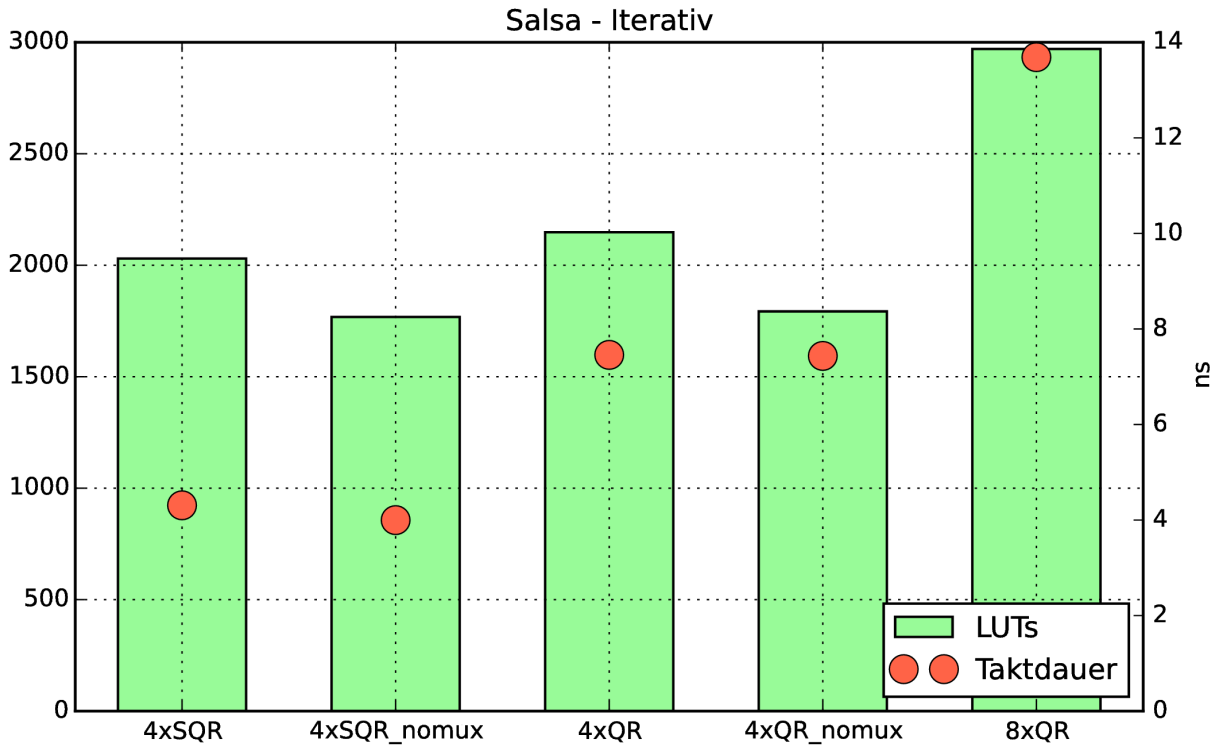
- SQR braucht 8 x 4-zu-1 32Bit Multiplexer
  - 4SQR → 32 x 4-zu-1 32Bit Multiplexer



03 Salsa
Ergebnisse

	Taktdauer (ns)	Frequenz (Mhz)	LUTs	FFs	LUTRAM	Durchsatz (Gbit/s)
4xSQR	4,308	232	2030	1034	-	3,71
4xSQR(opt)	3,998	250	1768	1032	-	4,00
4xQR	7,457	134	2148	1043	-	8,58
4xQR(opt)	7,435	134	1793	1034	-	8,61
8xQR	13,689	73	2970	1046	-	9,35
Pipe4	3,175	315	2742	2324	768	20,16
Pipe8	3,257	307	4316	3851	1280	39,30
Pipe16	3,264	306	7286	6920	2304	78,43
Pipe32	2,908	344	12928	13464	4480	176,06

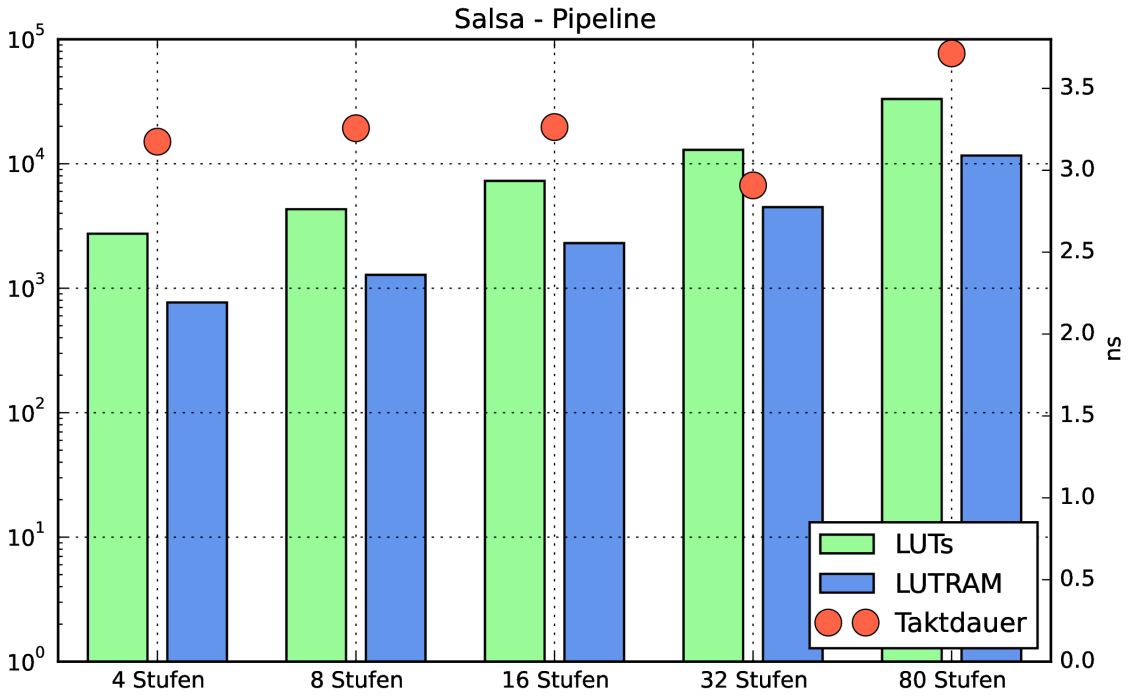
03 Salsa
Ergebnisse



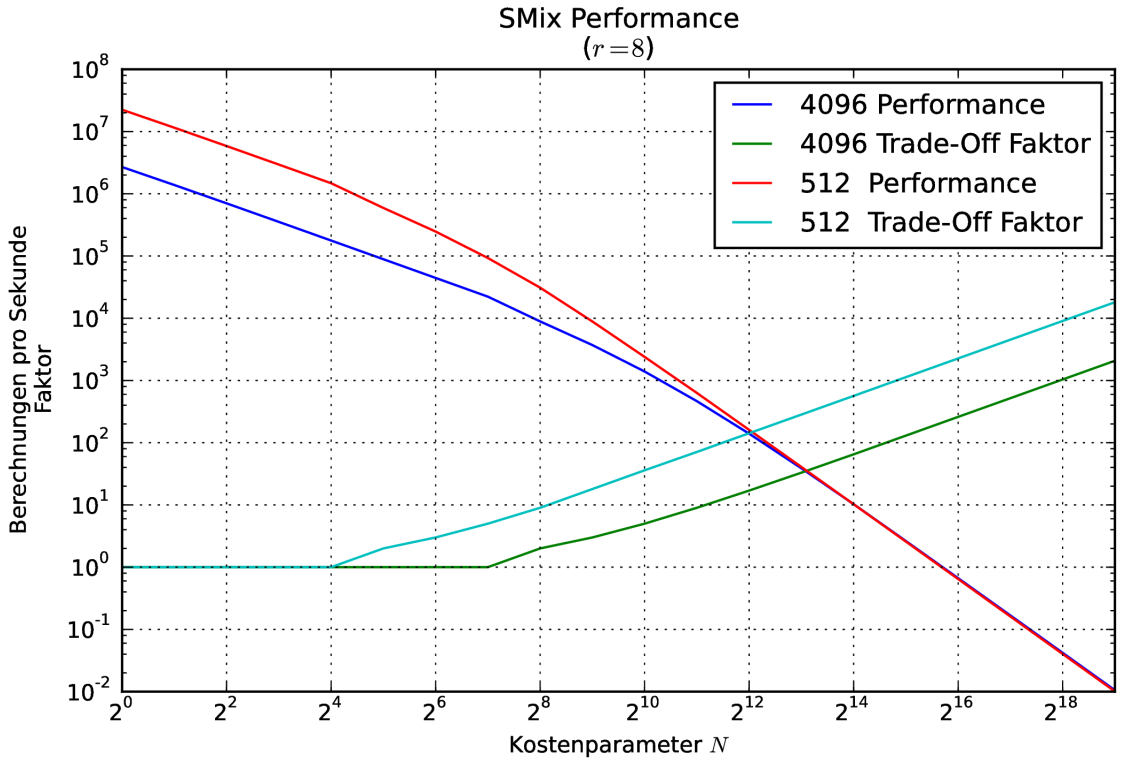
03 Salsa
Pipeline

- Yan [6] Pipeline: 4800 MBit/s, 470 kGE → 10,21 MBit/s/kGE
- Henzen [4] Iterativ: 6169 MBit/s, 24 kGE → 257 Mbit/s/kGE
- Yan [6] Iterativ: 255 MBit/s, 23 kGE → 11,09 MBit/s/kGE

03 Salsa
Ergebnisse



04 Entwurf & Abschätzung Scrypt
FPGA mit Block RAM



04 Entwurf & Abschätzung Scrypt
FPGA mit DDR3 RAM