Großer Beleg
Realisierung eines Turbo Decoders für die ARRIVE Architektur

Martin Zimmerling
mz793134@inf.tu-dresden.de

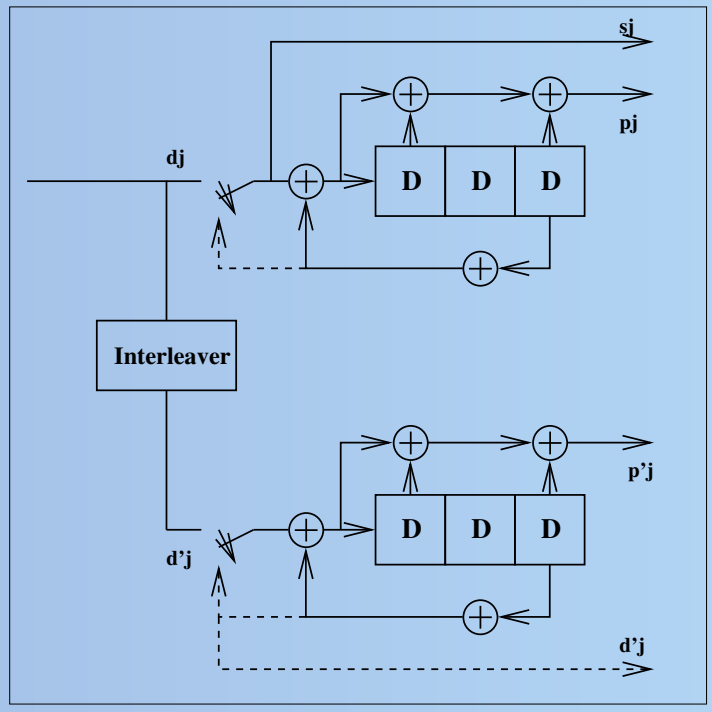
Gliederung

- Motivation
- Turbo-Codes
- Rekonfigurierbare Architekturen
- Implementierung
- Ergebnisse

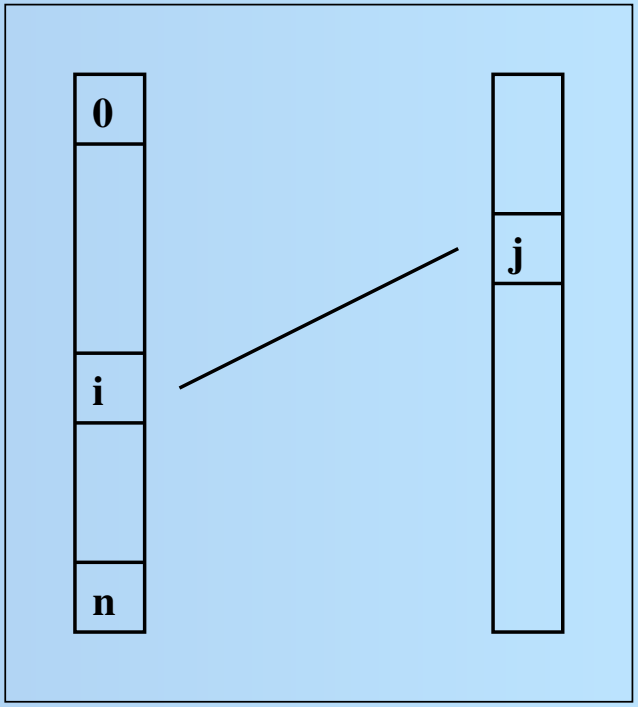
Motivation

- digitale Signalverarbeitung allgegenwärtig
- Bedarf an hohen Datenraten bei geringen Kosten
- Turbo Code als Beispielalgorithmus für digitale Signalverarbeitung
- effiziente Umsetzung auf rekonfigurierbaren Architekturen

1993: Berrou, Glavieux, Thitimajshima: „Near Shannon limit error-correcting coding and decoding: Turbo-codes“

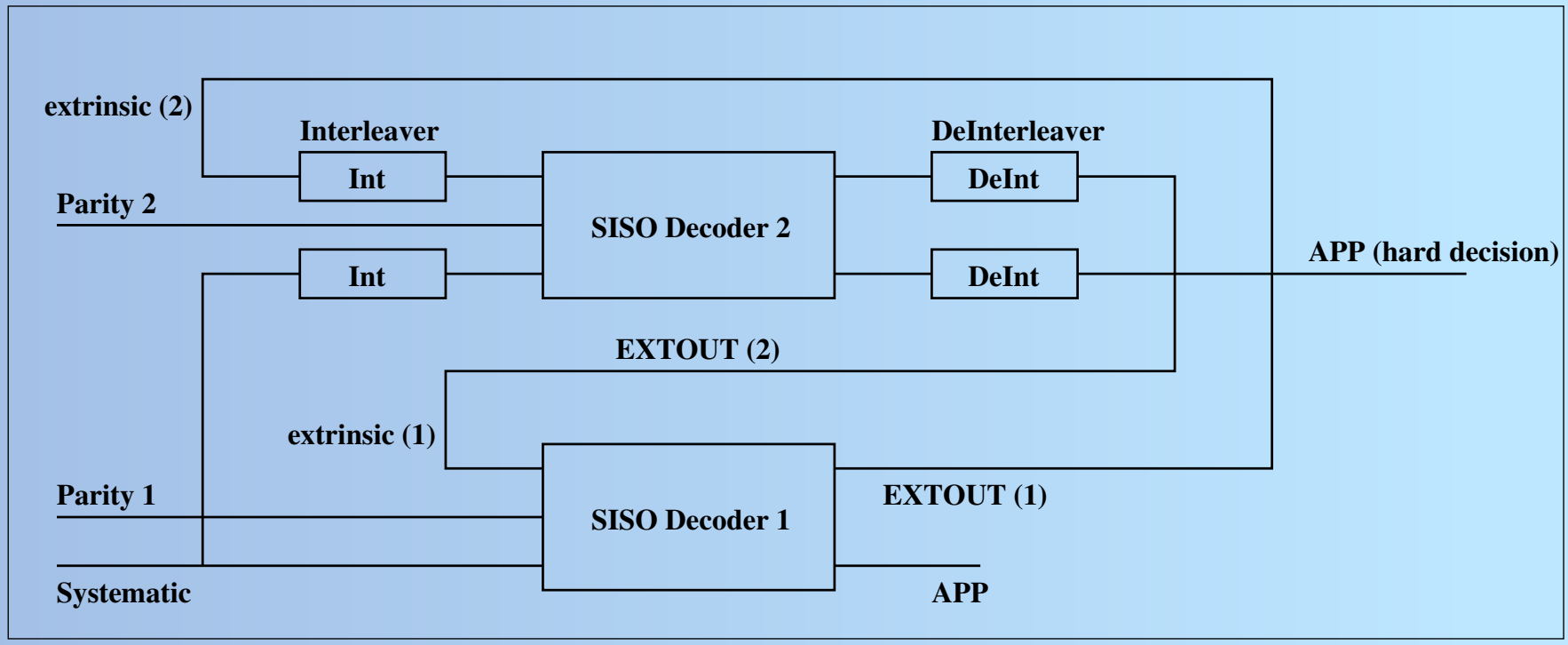


(a) Codierer



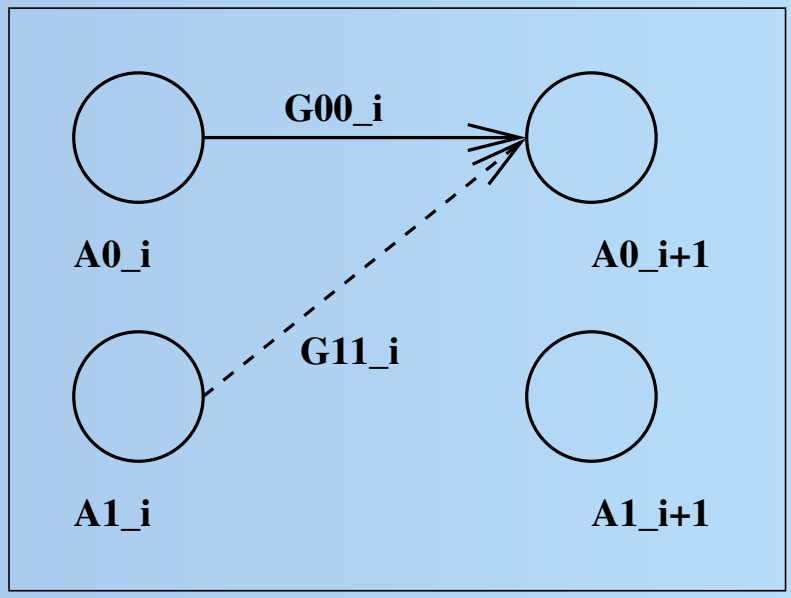
(b) Interleaver

Turbo-Decoder



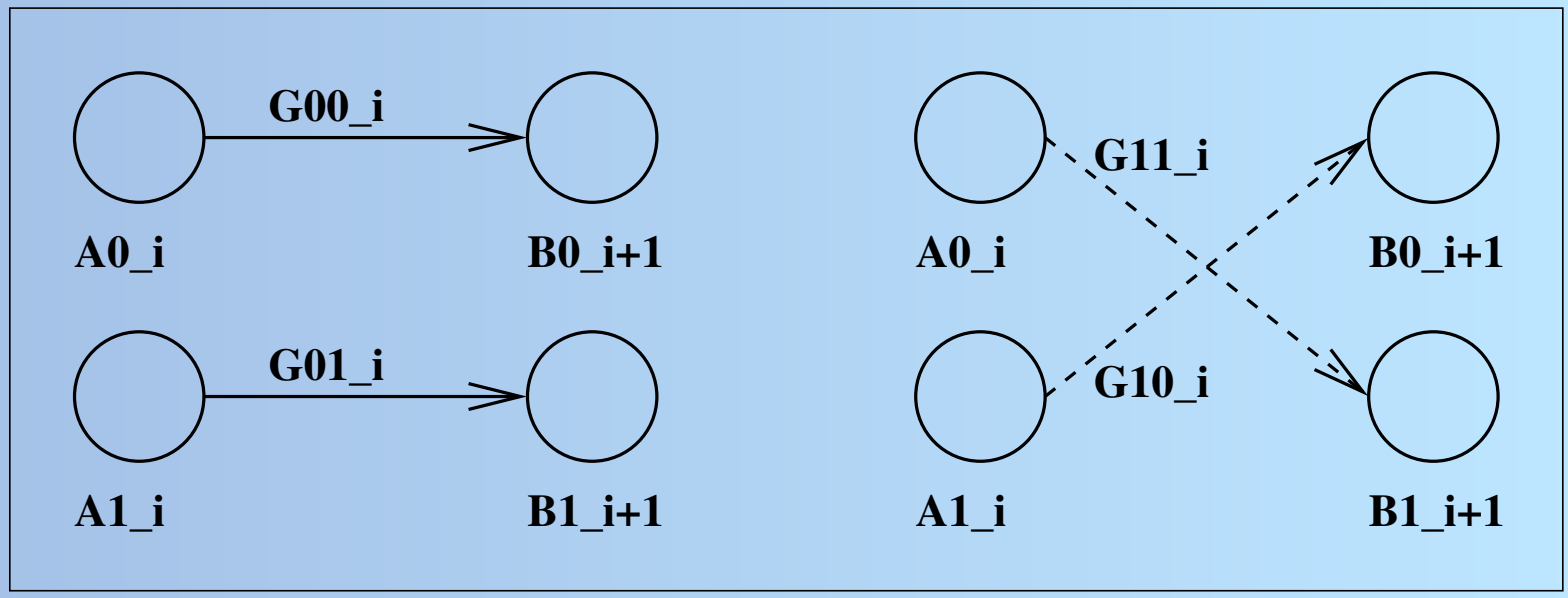
- Softinformation
- iterative Dekodierung

SISO-Decoder I



- Viterbi-Butterfly
  - $A0_{i+1} = max(A0_i + G00_i, A1_i - G00_i)$
  - $Ax_{i+1} = max(A1_i + G00_i, A0_i - G00_i)$
- vorwärts (Alpha), rückwärts (Beta)

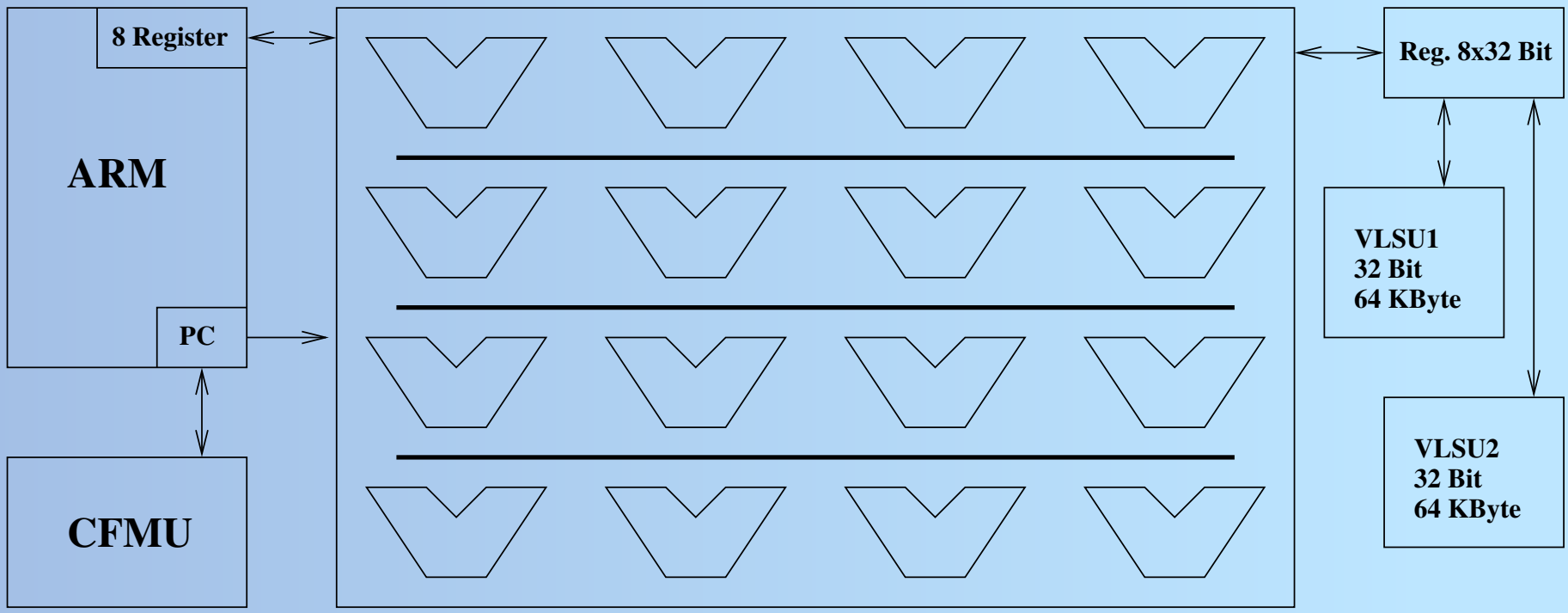
SISO-Decoder II



- Berechnung Maximum aller 0-Übergänge
- Berechnung Maximum aller 1-Übergänge

Rekonfigurierbare Architekturen

	Standard-Prozessor	ASIC	FPGA feingranular	FPGA grobgranular
Implementierungszeit	++	-	+	+ / ++
HW-Ausnutzung	-	++	-	+
Energieeffizienz	-	++	+	+
Veränderbarkeit des Entwurfs	++	-	+	+
Anpassbarkeit an versch. Alg.	++	-	0	++



- RALU: 4 x 4 ALUs 16 Bit
- 2 Vector Load/Store Units (32 Bit Zugriffsbreite)
- Control Flow Management Unit

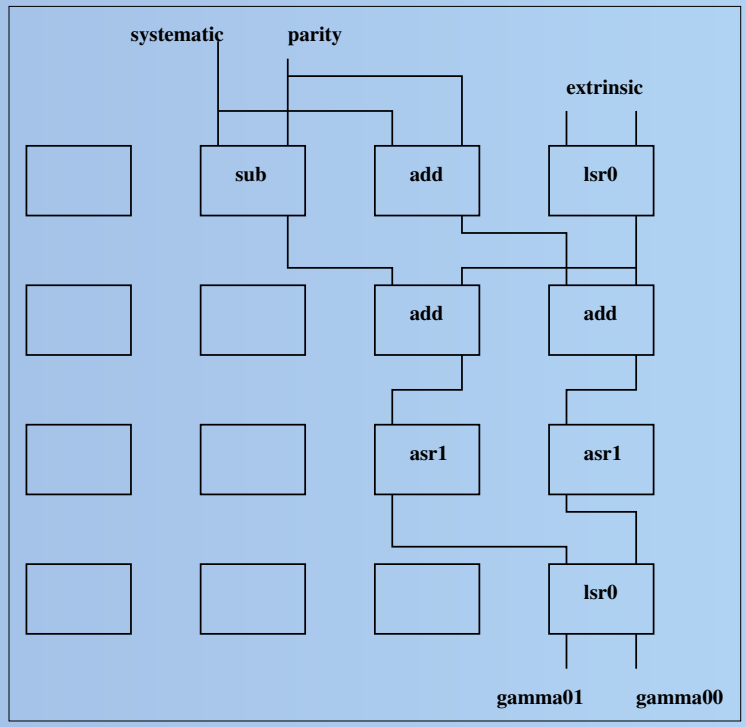
Implementierung

- C-Modell → ARM → ARRIVE
- Optimierung
  - 16 Bit Metriken (unsigned)
  - 3GPP-Standard (UMTS)
  - nur 1020 Bit maximale Paketgröße

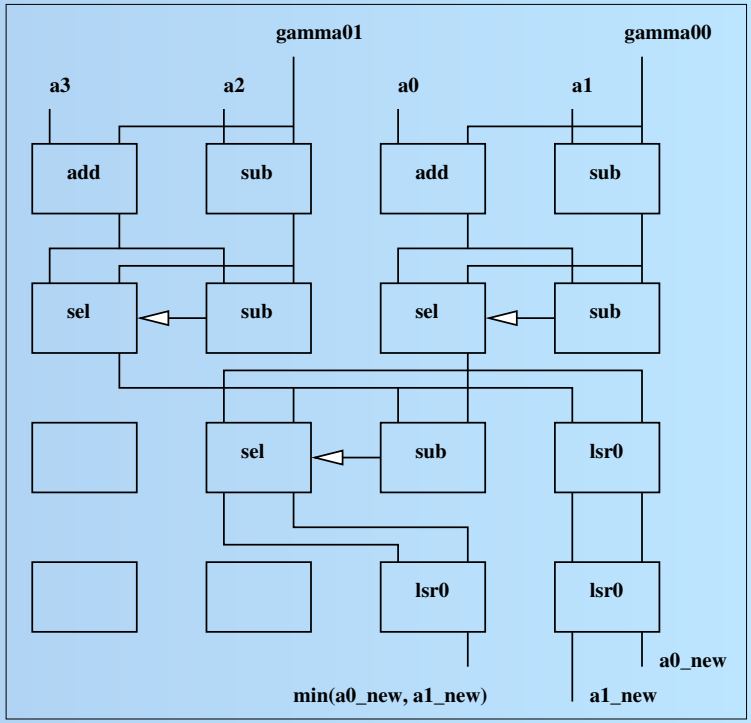
Alpha (Schleife)

Berechnung RALU	VLSU1	VLSU2
Gamma00[i], Gamma01[i]		
A[0, i], A[1, i], min(A[0, i], A[1, i])	speicher Gamma	
A[2, i], A[3, i], min(A[2, i], A[3, i])	speicher Alpha(0, 1)	
A[4, i], A[5, i], min(A[4, i], A[5, i])	speicher Alpha(2, 3)	
A[6, i], A[7, i], min(A[6, i], A[7, i])	speicher Alpha(4, 5)	
Minimum A[z, i] - 0x1000	speicher Alpha(6, 7)	
Subtraktion Minimum Alpha 0 - 3		lade *systematic*, *parity*
Subtraktion Minimum Alpha 4 - 7		lade *extrinsic*

Alpha (RALU)



(a) Gamma

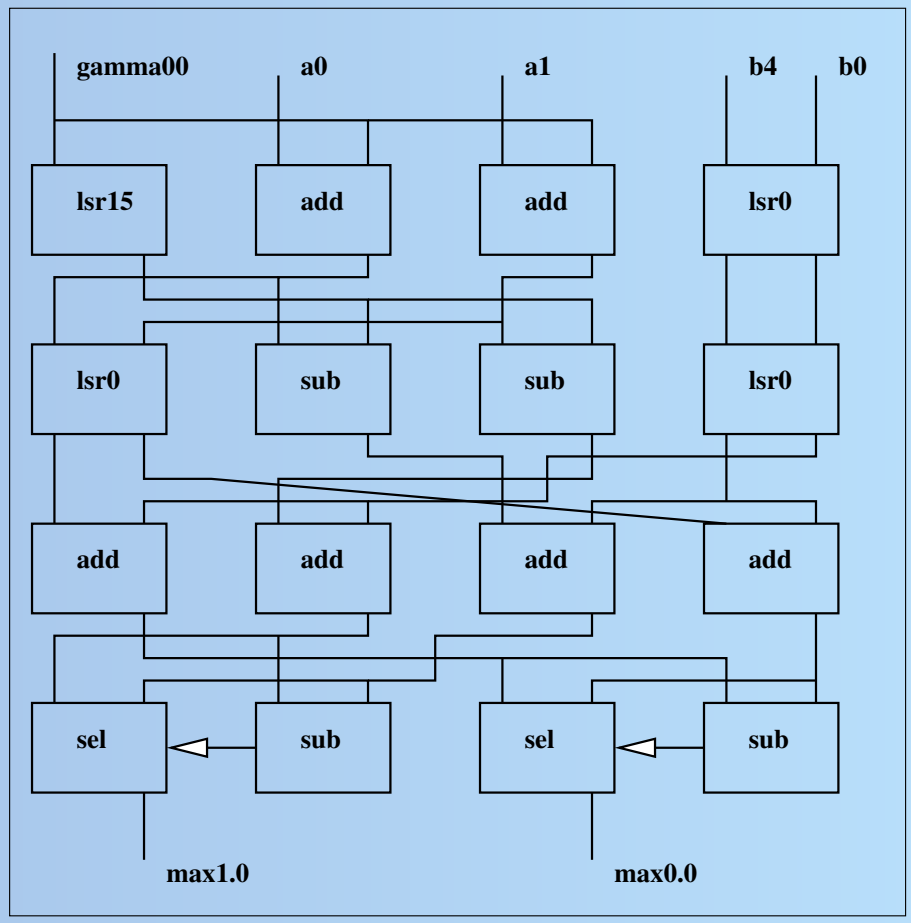


(b) Alpha

APP (Schleife)

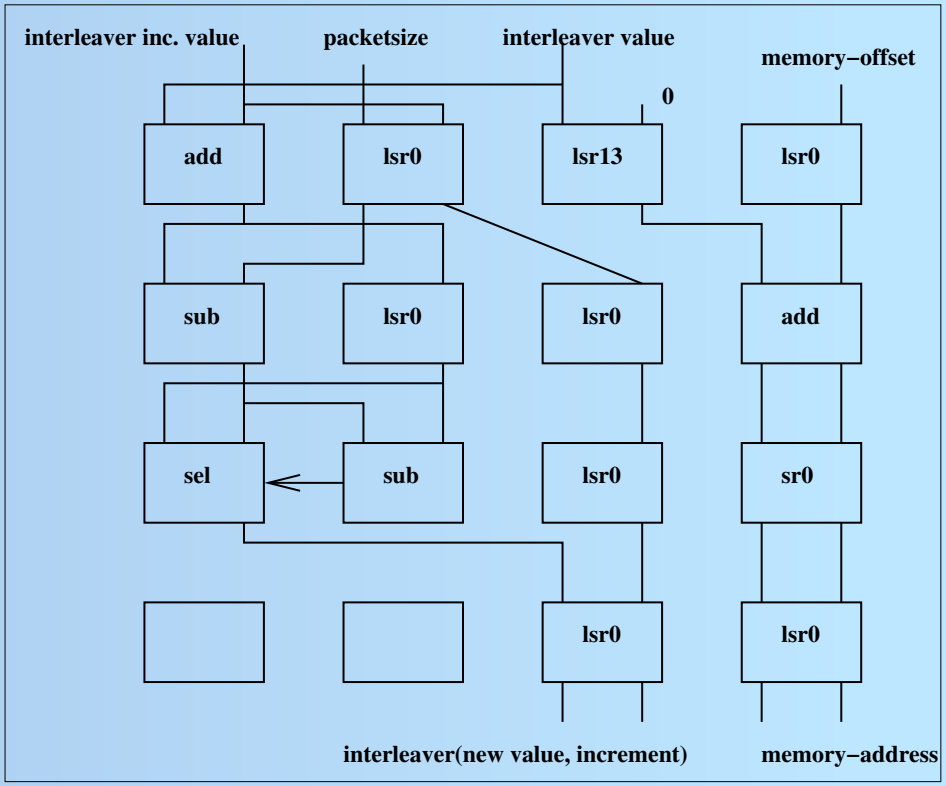
Berechnung RALU	VLSU1	VLSU2
	lade Alpha(0, 1)	lade Beta(0, 4)
(max0, max1).0	lade Alpha(2, 3)	lade Beta(1, 5)
(max0, max1).1	lade Alpha(4, 5)	lade Beta(2, 6)
(max0, max1).2	lade Alpha(6, 7)	lade Beta(3, 7)
(max0, max1).3		
max0		lade systematic, parity
max1		lade extrinsic
APP	lade Gamma[i+1]	
EXTOUT		speicher APP
		speicher EXTOUT

APP (RALU)

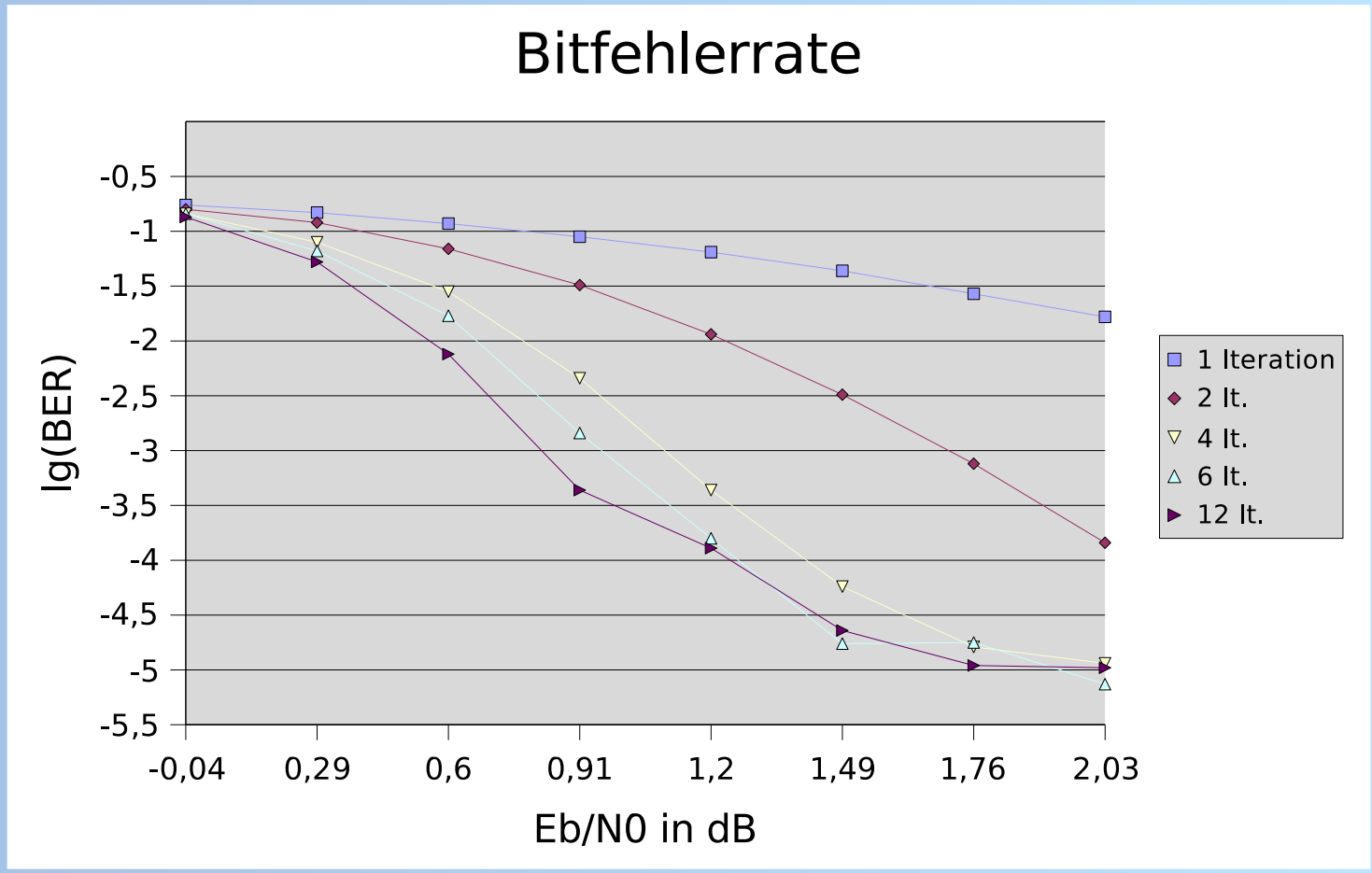

gamma00
a0
a1
b4
b0
lsr15
add
add
lsr0
lsr0
sub
sub
lsr0
add
add
add
add
sel
sub
sel
sub
max1.0
max0.0

Interleaver

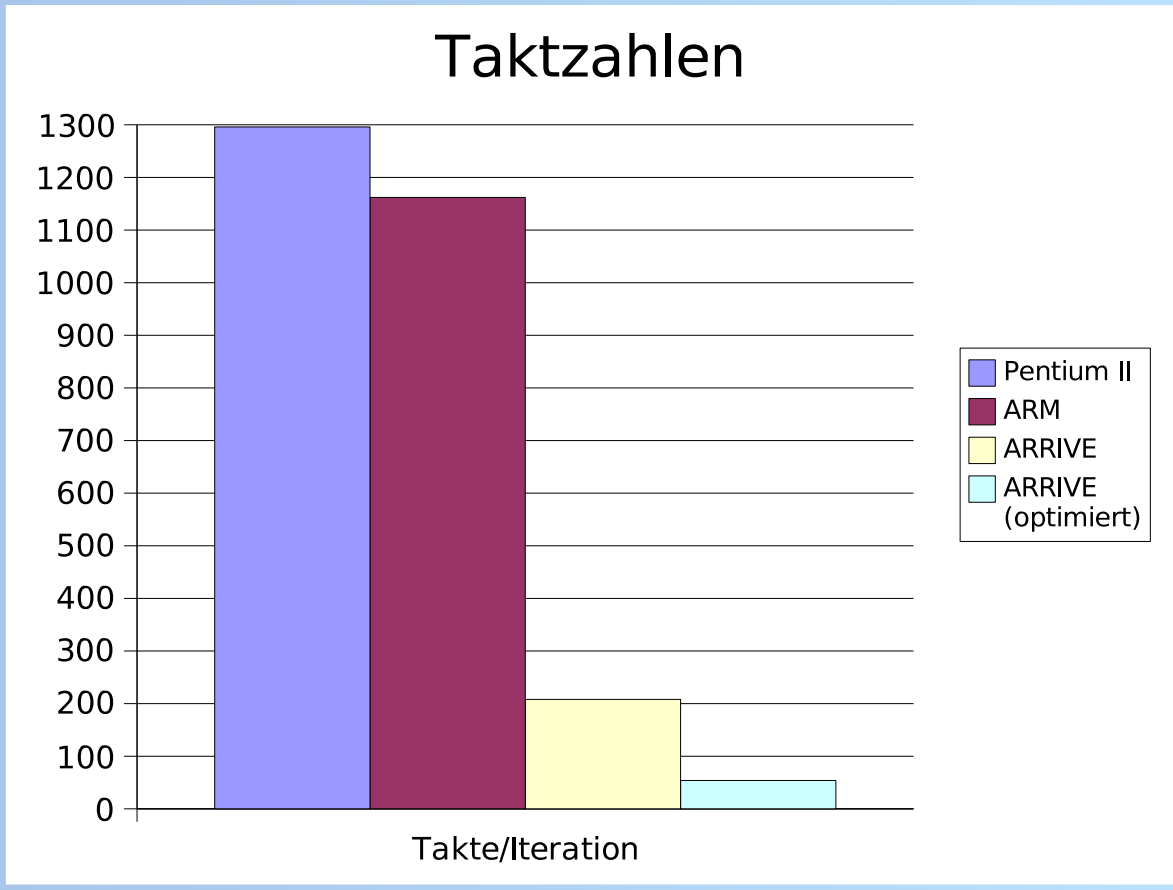
VLSU1	VLSU2	RALU
	lade Wert	Adresse
	speicher W.	



- relative-prime Interleaver
- Interleaver: on-the-fly vs. speicherbasiert


Bitfehlerrate
lg(BER)
-0,5
-1
-1,5
-2
-2,5
-3
-3,5
-4
-4,5
-5
-5,5
-0,04
0,29
0,6
0,91
1,2
1,49
1,76
2,03
Eb/N0 in dB
1 Iteration
2 It.
4 It.
6 It.
12 It.

Geschwindigkeit



Ausnutzung

- 27 Takte / halbe Iteration, Informationsbit
- VLSU:
  - 11 x VLSU1 (41 %)
  - 16 x VLSU2 (59 %)
  - gesamt 50 %
- 202 16 Bit Operationen (47 %)
  - 140 x Add/Sub (69 %)
  - 45 x Sel (22 %)
- Speicherausnutzung (2 x 64 KByte): >50 %

Zusammenfassung

- Implementierung folgt 3GPP Standard
- geringe Änderungen an Architektur notwendig
- im Vergleich zu ARM sehr effizient