TECHNISCHE UNIVERSITÄT DRESDEN

Verteidigung der Studienarbeit

ANALYSE DER BEOBACHTBARKEIT DER FEHLERFORTPFLANZUNG IN EINEM MIKROPROZESSOR ANHAND VERSCHIEDENER TRACE-KONFIGURATIONEN

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Dresden, 11.05.2017
Matthias Brinker

Gliederung

Einleitung und Motivation

Problem:

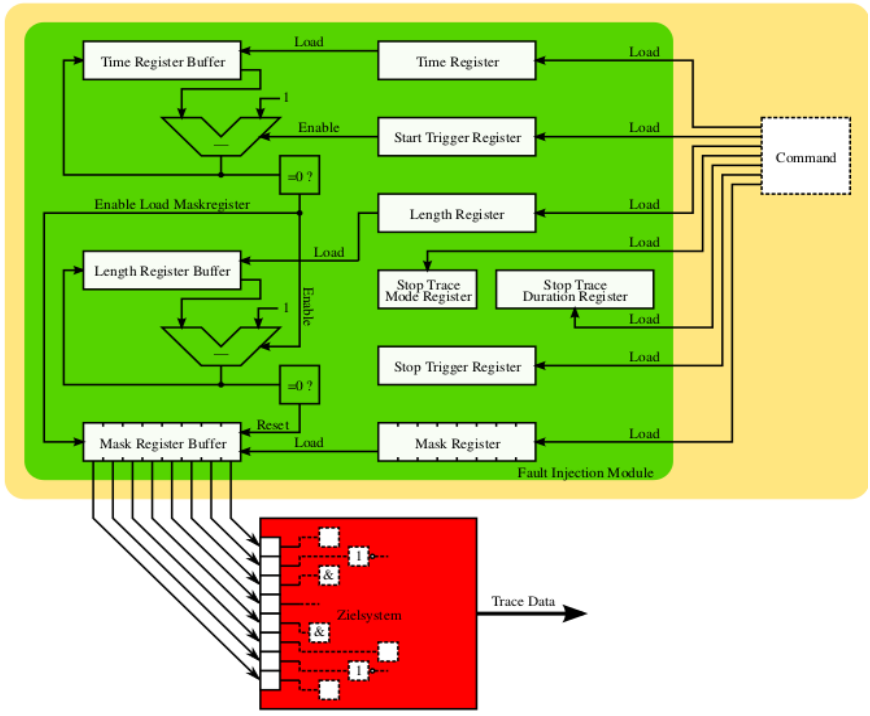
- Komplexere, kleinere Bauteile → anfälliger für Störung
- Eingebundene Systeme nicht anhaltbar

Lösung: Trace und Instrumentierungsbasierte Hardwarefehlerinjektion

- Relevante Daten werden aufgezeichnet
- Auswertung der Daten nicht zur Laufzeit
- Fehler werden mit Hilfe der Trace-Hardware injiziert
- Zwei Ansätze (Fidalgo und Gunia)

Einleitung und Motivation

Fehlerinjektion nach Gunia



Quelle: [1]

Time Register:
Zeitpunkt der Injektion

Length Register:
(zeitliche) Länge der Injektion

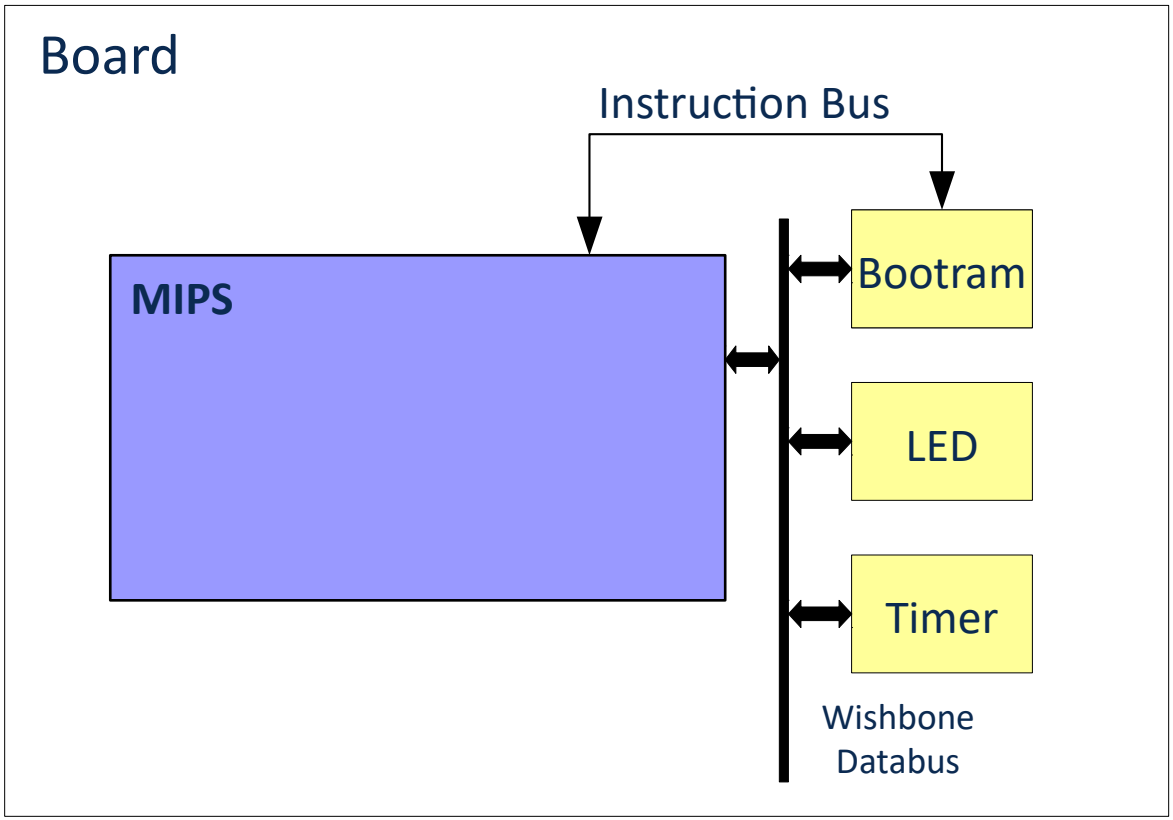
Mask Register:
Ort der Injektion

Entwurf und Implementierung

Endwurf und Implementierung

Board:

- Xilinx ML505
- MIPS-Prozessor
- 6,25 MHz



Implementierung

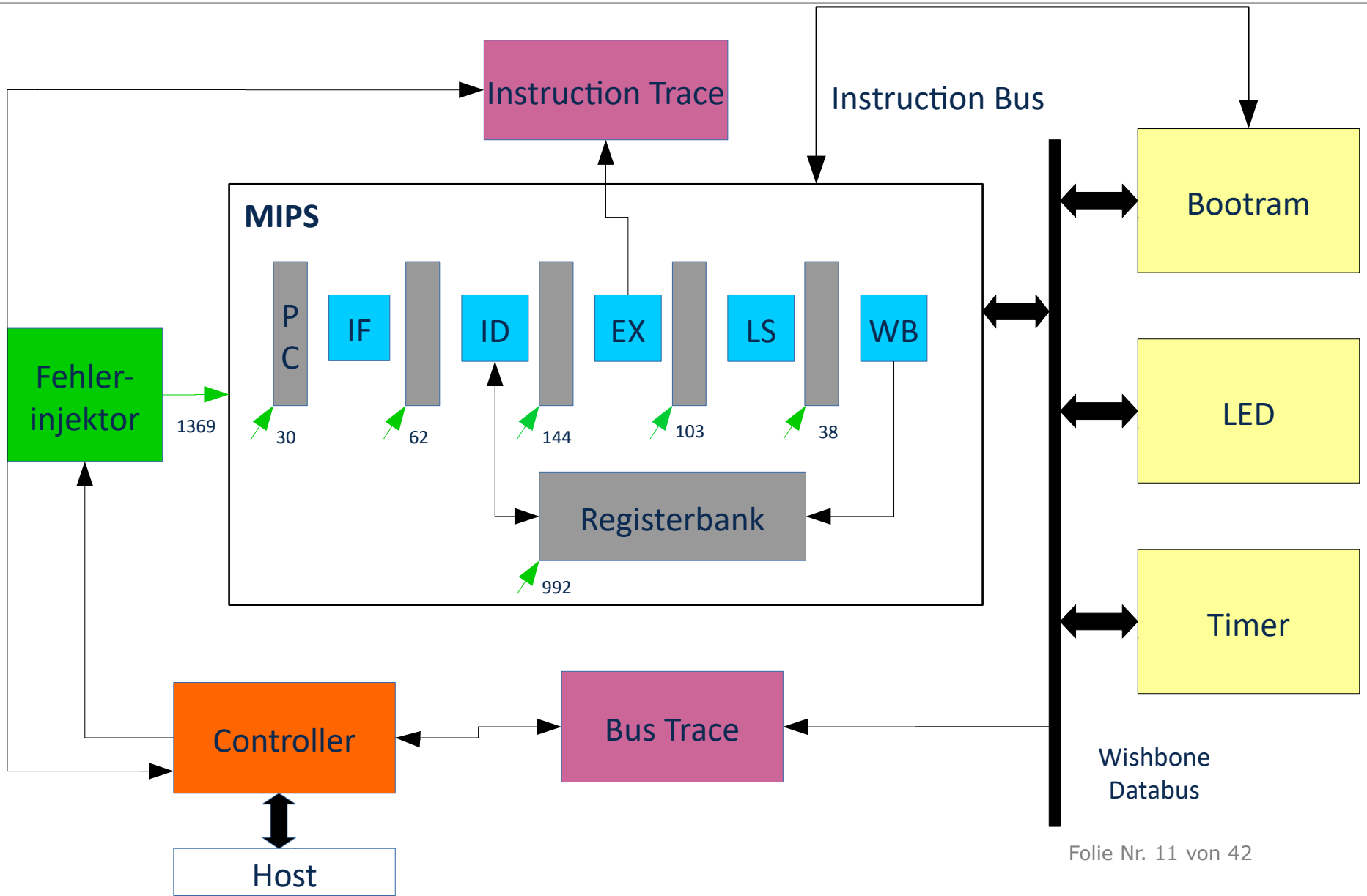
Programme:

- Fibonacci-Folge (Fibo)
- Primzahlen Sieb (PZ)
- Quicksort (QS)

Endwurf und Implementierung

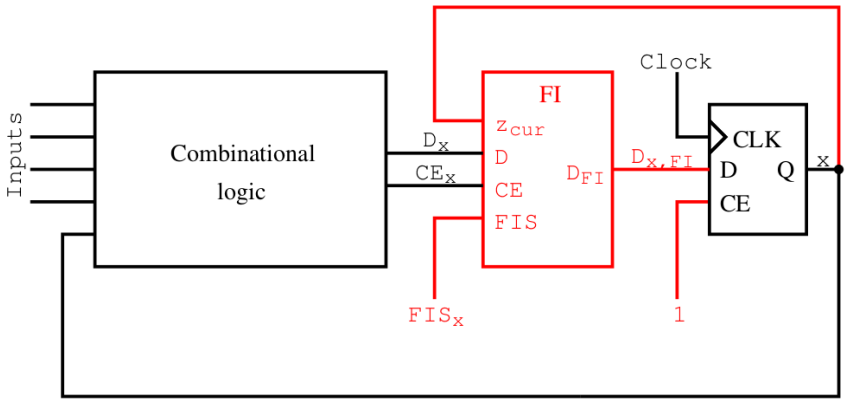
Programme:

Programm	Laufzeit (Takte)	Komplexität	Registerbank Register
Fibo	174	einfach	6/31
PZ	7216	moderat	11/31
QS	645	komplex	17/31


Instruction Trace
Instruction Bus
MIPS
P C
IF
ID
EX
LS
WB
30
62
144
103
38
Registerbank
992
Fehler-injektor
1369
Bootram
LED
Timer
Controller
Bus Trace
Host
Wishbone Databus

Endwurf und Implementierung

Fehlerinjektion vor Flip-Flops:



- Z_{cur} : aktuelle Wert des Flip-Flops
- FIS_x : Wert von Controller, ob Fehlerinjektion ausgeführt werden soll

Quelle: [1]

Endwurf und Implementierung

Fehlermodelle:

Fehler in Speichergliedern

- Flip-to-0 $(\overline{FIS} \cdot zcur) \cdot \overline{CE} + (\overline{FIS} \cdot D) \cdot CE$
- Flip-to-1 $(FIS + zcur) \cdot \overline{CE} + (FIS + D) \cdot CE$
- Bit-Flip $(\overline{FIS} \otimes zcur) \cdot \overline{CE} + (\overline{FIS} \otimes D) \cdot CE$

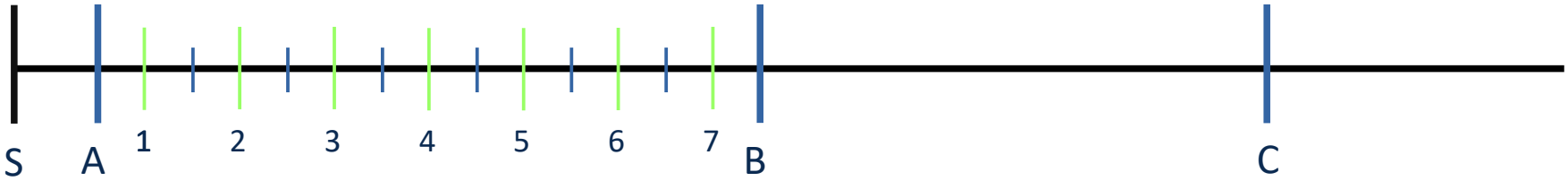
Testbedingungen

Testbedingungen

Kampagne

- Bestimmtes Programm und Fehlermodell
  → 9 Kampagnen (3 Fehlermodelle, 3 Programme)
- Fehlerinjektion in alle benutzen Register
- Deterministischen Zeitpunkten (Äquidistant)

Testbedingungen



S : Programmstart
A : Start Trigger
B : Stop Trigger
C : Trace wird von Host-PC gestoppt (nach 1 Sekunde)

: Fehlerinjektionszeitpunkte

Auswertungsverfahren

Auswertungsverfahren

Auswertung:

- „Golden Run“ als Referenz
- „Golden Run“ sind die Trace-Daten eines fehlerfreien Durchlaufs
- Vier verschiedene Szenarien

Golden Run

Befehl	Daten
Jump	38
Load	A:70
Store	A:70 D:5
Jump	4c
Load	A:74
Store	A:74 D:6
Jump	4c
Store	A:70 D:7
Jump	Endlos

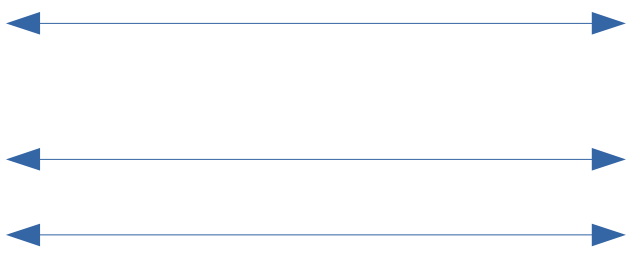
Auswertungsverfahren

Kein Fehler

Szenario 1

Golden Run

Befehl	Daten
Jump	38
Load	A:70
Store	A:70 D:5
Jump	4c
Load	A:74
Store	A:74 D:6
Jump	4c
Store	A:70 D:7
Jump	Endlos



Faulty Run

Befehl	Daten
Jump	38
Load	A:71
Store	A:70 D:5
Jump	4c
Load	A:74
Store	A:74 D:6
Jump	4c
Store	A:70 D:7
Jump	Endlos

Auswertungsverfahren

Szenario 2

Golden Run

Befehl	Daten
Jump	38
Load	A:70
Store	A:70 D:5
Jump	4c
Load	A:74
Store	A:74 D:6
Jump	4c
Store	A:70 D:7
Jump	Endlos

Fehler

Faulty Run

Befehl	Daten
Jump	38
Load	A:71
Store	A:70 D:5
Jump	4c
Load	A:74
Store	A:74 D:6
Jump	4c
Store	A:70 D:7
Jump	Endlos

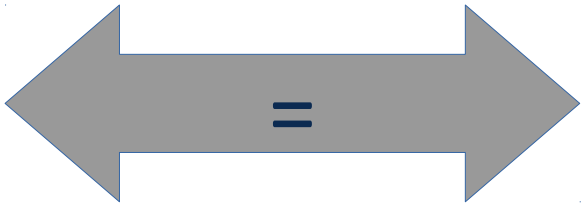
Auswertungsverfahren

Kein Fehler

Szenario 3

Golden Run

Befehl	Daten
Jump	38
Load	A:70
Store	A:70 D:5
Jump	4c
Load	A:74
Store	A:74 D:6
Jump	4c
Store	A:70 D:7
Jump	Endlos



Faulty Run

Befehl	Daten
Jump	38
Load	A:71
Store	A:70 D:5
Jump	4c
Load	A:74
Store	A:74 D:6
Jump	4c
Store	A:70 D:7
Jump	Endlos

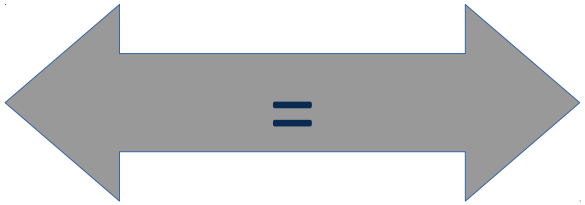
Auswertungsverfahren

Kein Fehler

Szenario 4

Golden Run

Befehl	Daten
Jump	38
Load	A:70
Store	A:70 D:5
Jump	4c
Load	A:74
Store	A:74 D:6
Jump	4c
Store	A:70 D:7
Jump	Endlos



Faulty Run

Befehl	Daten
Jump	38
Load	A:71
Store	A:70 D:5
Jump	4c
Load	A:74
Store	A:74 D:6
Jump	4c
Store	A:70 D:7
Jump	Endlos

Auswertungsverfahren

Szenarien

Szenario	Vorteile	Nachteile
1 : Final State	• Keine unentdeckten Fehler • Keine falsch positiv Ergebnisse	• Benötigt ganzen Trace
2 : Instr + Data	• Keine unentdeckten Fehler	• Benötigt ganzen Trace
3 : Instr	• Wenig Trace-Daten	• Unentdeckte und falsch Positive Fehler
4 : Data	• Weniger Trace-Daten	• Unentdeckte und falsch Positive Fehler

Auswertungsverfahren

Bewertungsformeln

- k(i,t) : Entscheidungsformel für einen Ort i und Zeitpunkt t
- R(i) : Durchschnittliche Fehlerwahrscheinlichkeit an einem Ort i
- $\overline{R}$: Durchschnittliche Fehlerwahrscheinlichkeit

$$k(i,t) = \begin{cases} 0 & \text{wenn falsch} \\ 1 & \text{wenn korrekt} \end{cases}$$

$$R(i) = 1 - \frac{1}{T}\sum_{t=1}^{T} k(i,t)$$

$$\overline{R} = \frac{1}{N}\sum_{i=1}^{N} R(i)$$

Analyse

Analyse

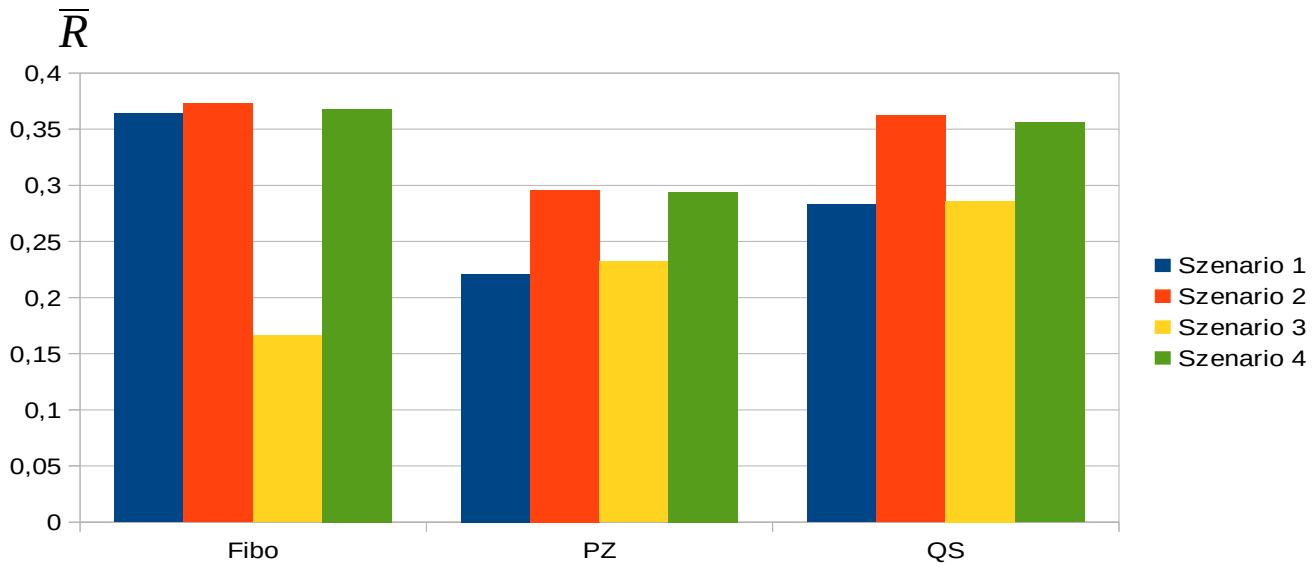
Bit-Flip Fehlerinjektion

- Bit-Flip Fehler höhste Fehlerwahrscheinlichkeit von allen Fehlermodellen
- (theoretisch) wenn in Flip-to-1 oder Flip-to-0 ein Fehler, dann auch in Bit-Flip

$$k^{\text{Bit-Flip}}(i,t) = \begin{cases} 0 & \text{wenn } k^{\text{Flip-to-1}}(i,t) = 0 \ \vee \ k^{\text{Flip-to-0}}(i,t) = 0 \\ 1 & sonst \end{cases}$$

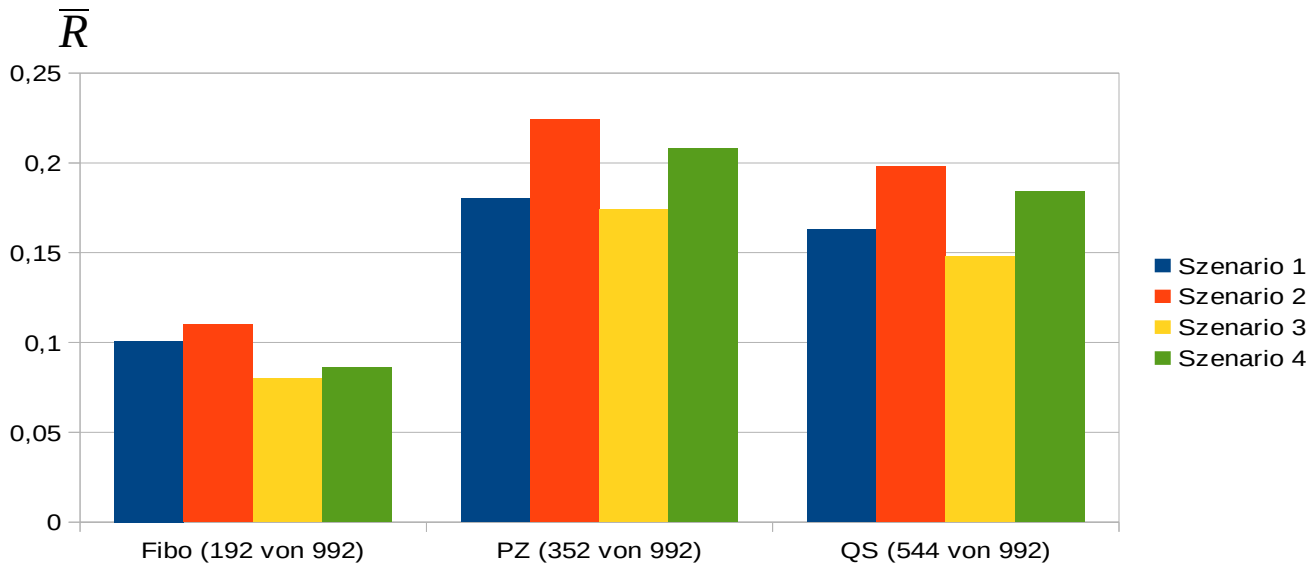
Analyse

Durchschnittliche Fehlerwahrscheinlichkeiten (Pipeline)

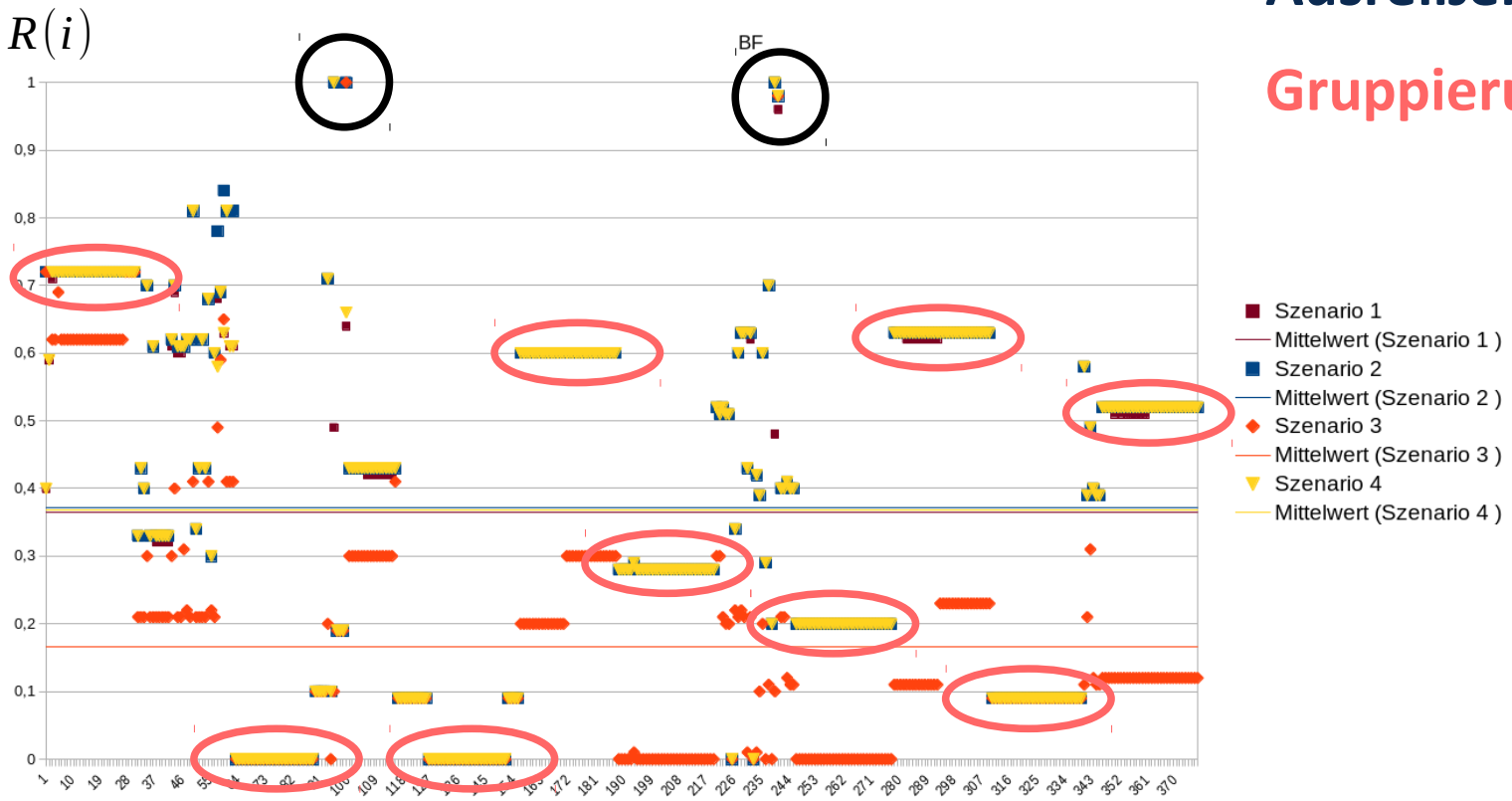


Analyse

Durchschnittliche Fehlerwahrscheinlichkeiten (Registerbank)



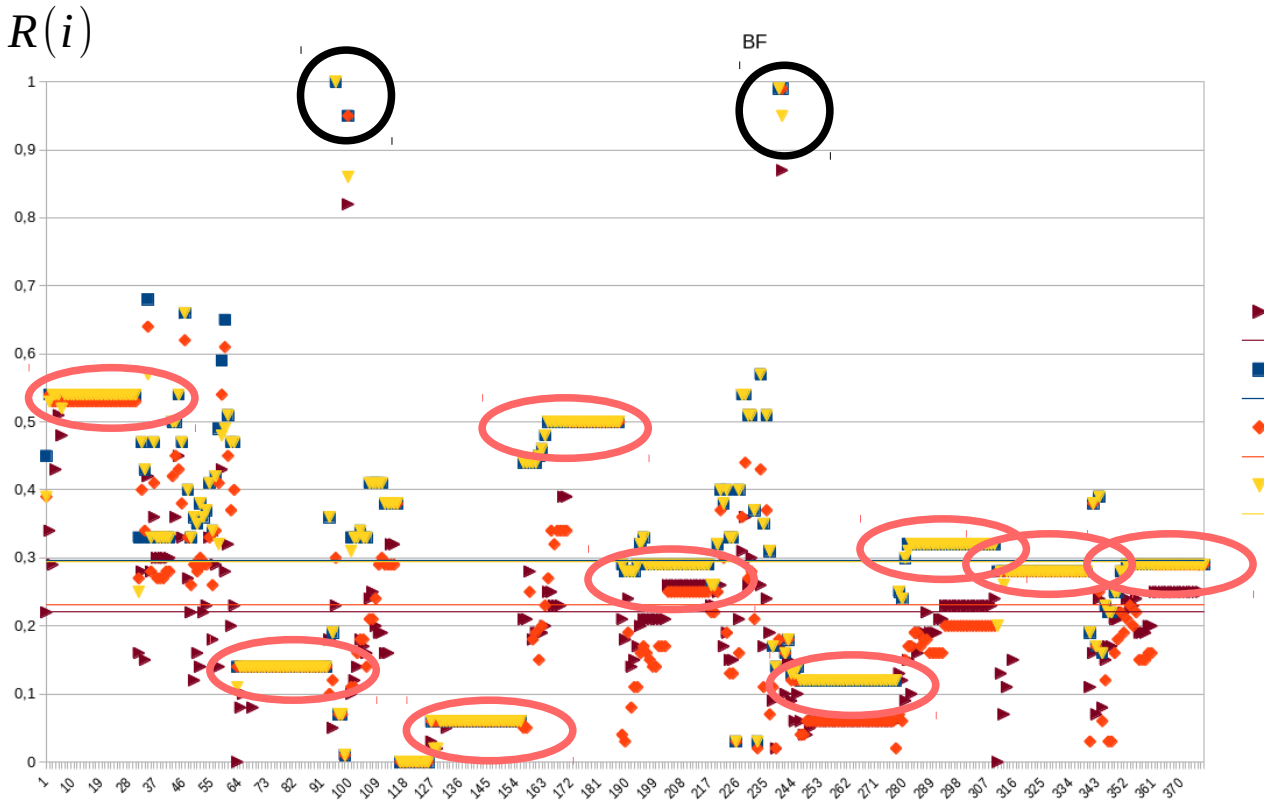
Bit-Flip Fehlerinjektion in Fibo (Pipeline Register)



Ausreißer

Gruppierungen

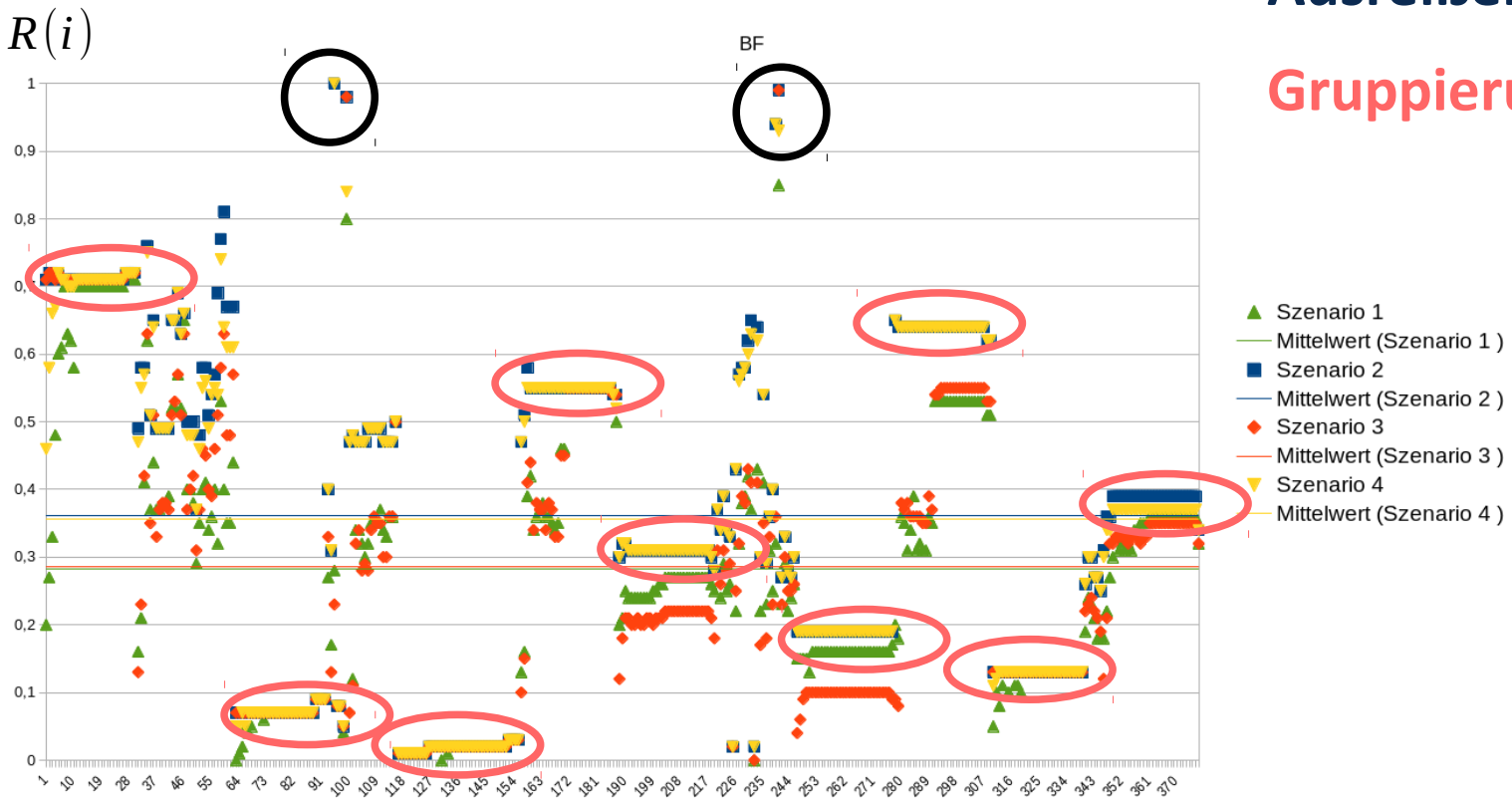
Bit-Flip Fehlerinjektion in PZ (Pipeline Register)



Ausreißer

Gruppierungen

Bit-Flip Fehlerinjektion in QS (Pipeline-Register)



Ausreißer

Gruppierungen

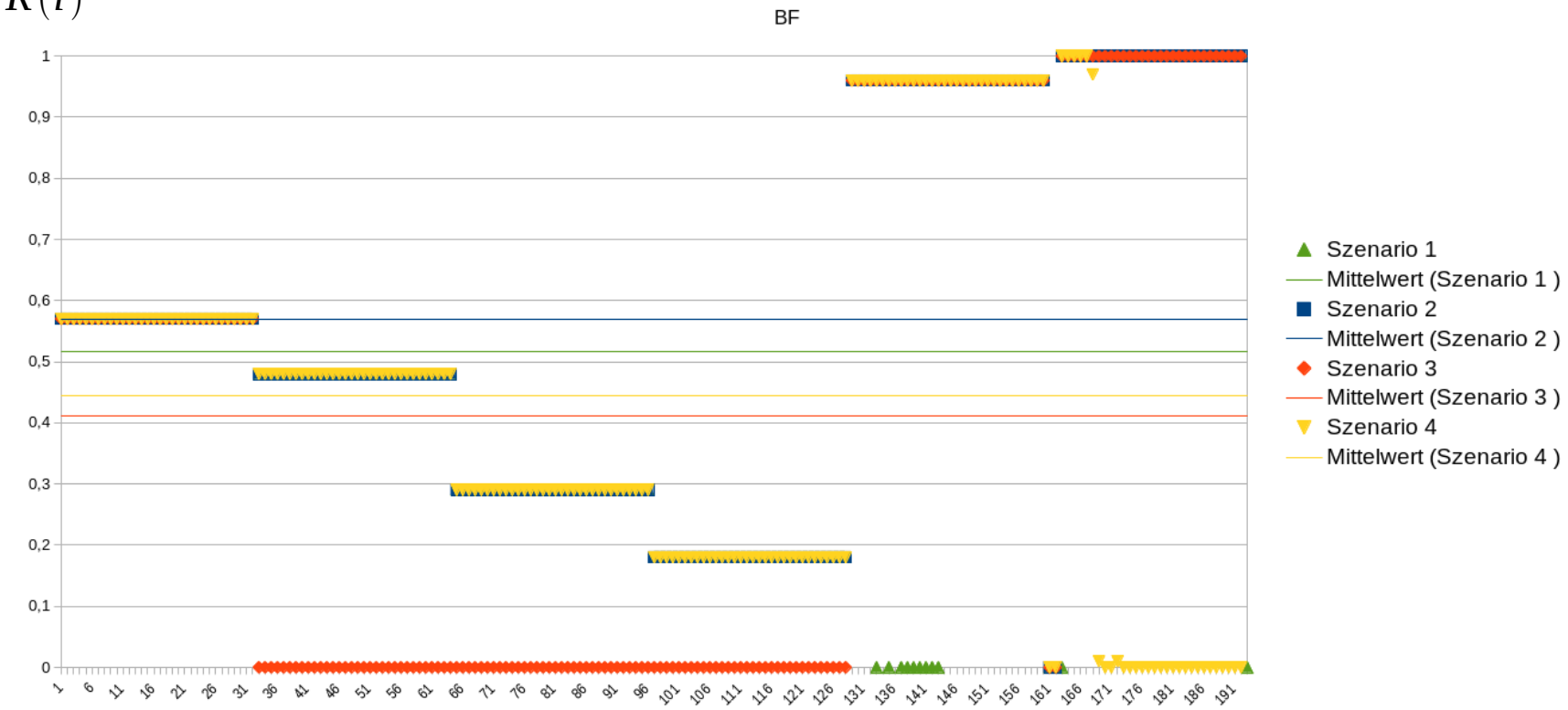
Analyse

Load/Store und Jump Commands

- Bit 94 und 238 Load/Store
- Bit 98 und 239 Jump Befehle

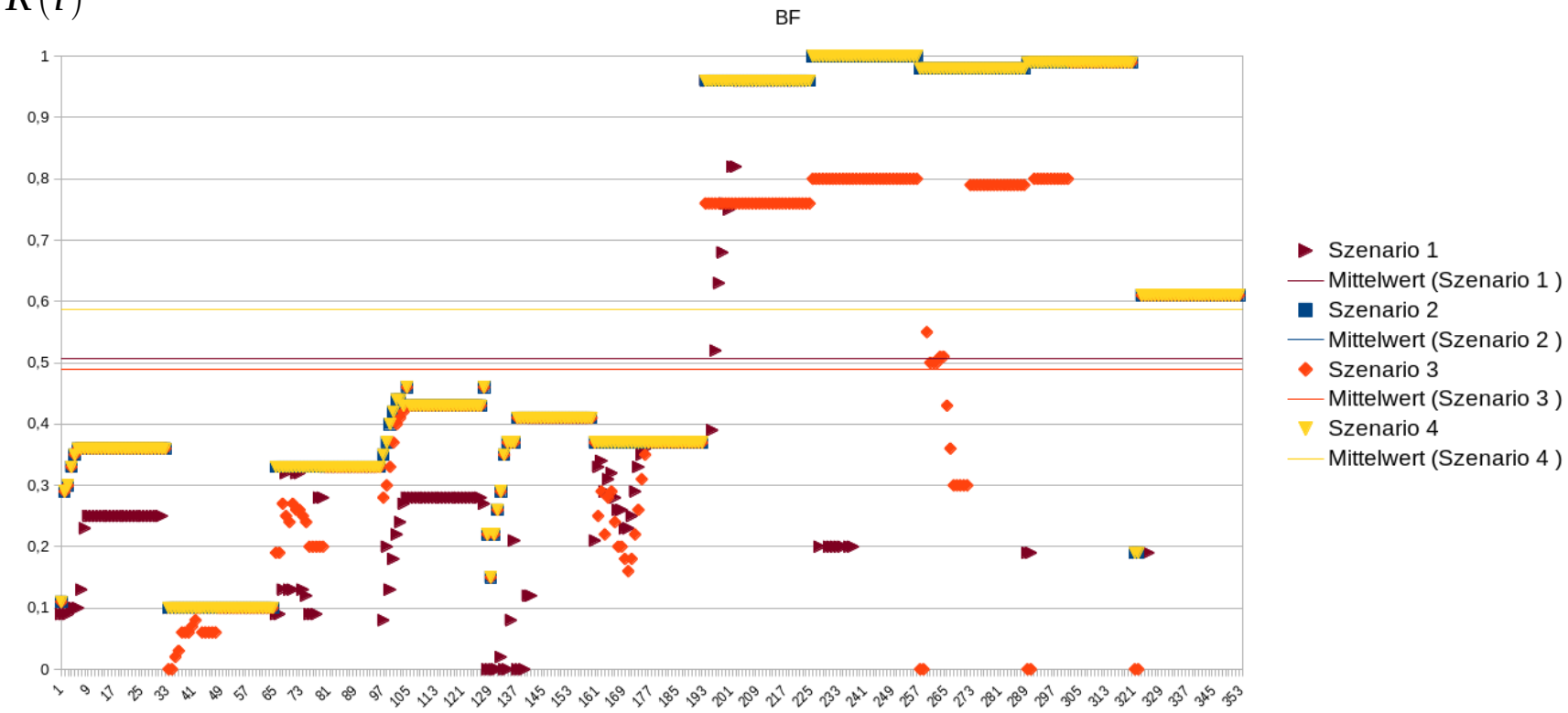
Bit-Flip Fehlerinjektion in Fibo (Registerbank)

$R(i)$



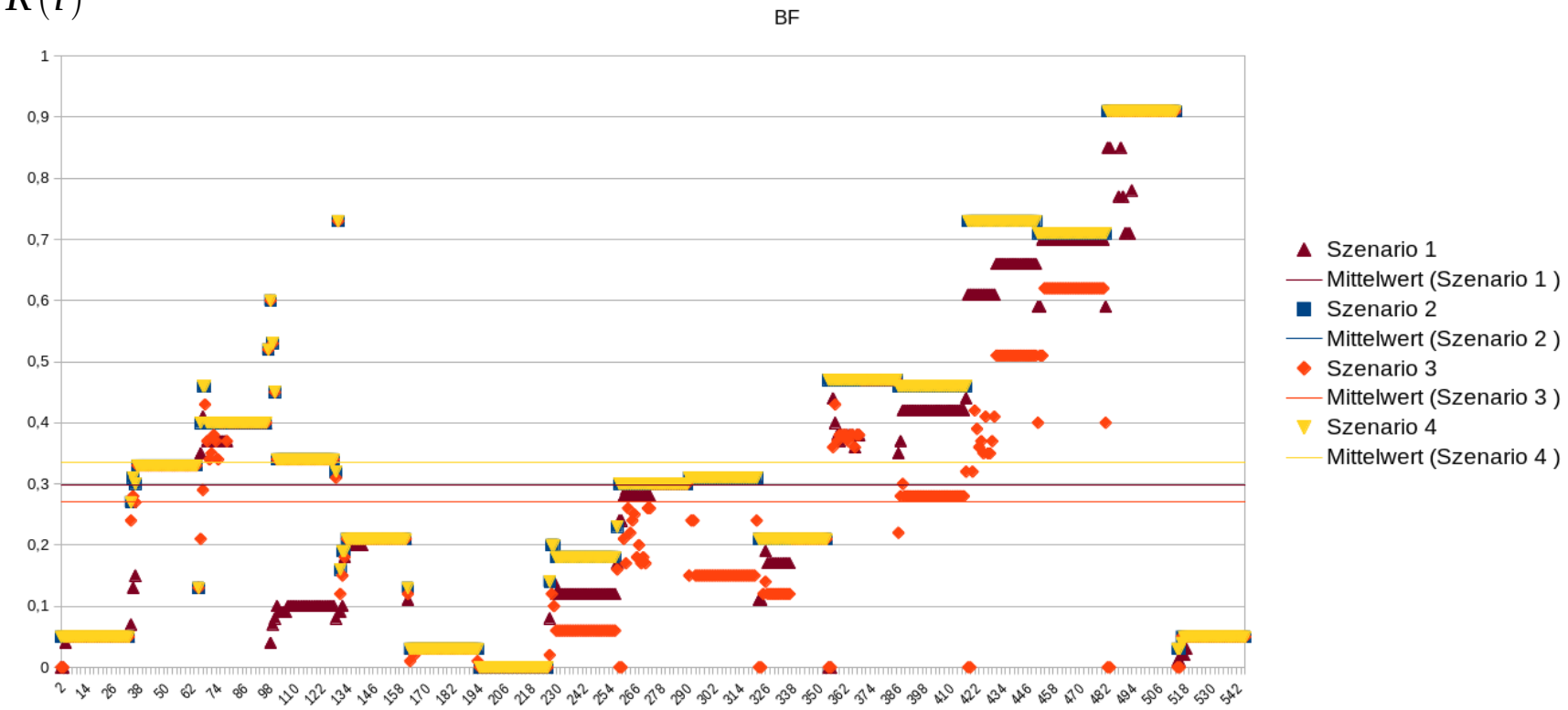
Bit-Flip Fehlerinjektion in PZ (Registerbank)

$R(i)$



Bit-Flip Fehlerinjektion in QS (Registerbank)

$R(i)$



Analyse

Bit-Flip Beispiel

- QS Durchlauf 2312
- Fehler in Flip-Flop 49
- Takt 39 (move-Befehl)

Analyse

Bit-Flip Beispiel

- move = add $d, $s, $t
- $t = 0-Register
- Bit-Flip in Bit 19 des ADD-Befehls
  → $t = $a2

ADD:

OP	rs	rt	rd	shamt	funct
$_{31}000000_{26}$	$_{25}$ sssss$_{21}$	$_{20}$ t t ttt$_{16}$	$_{15}$ddddd$_{11}$	$_{10}$- - - - -$_{6}$	$_{5}100000_{0}$

Zusammenfassung und Ausblick

Zusammenfassung

Auswertung

- 3 Programme, 3 Fehlermodelle, 4 Auswertungszenarien
- Fehlerinjektion zu 100 Zeitpunkten in 569 - 921 Flip-Flops
- Mehr als 650.000 ausgewertete Durchläufe

Zusammenfassung

Ergebnisse

- Fehlerwahrscheinlichkeit hängt stark von Fehlermodell und dem ausgeführten Programm ab
- Szenario 1 eignet sich am Besten für Fehlererkennung, allerdings viele Trace-Daten
- Durchläufe können sich in seltenen Fällen untereinander beeinflussen

Nächsten Schritte

- Trace-Zeitpunkte überarbeiten
- Prozessor nach jedem Durchlauf zurück setzen
- Weiter Analysen

Quellen

1) Erweiterung der Trace-Hardware eines Mikroprozessors für die Fehlerinjektion und Fehlerbeobachtung Marco Gunia

weitere Quellen siehe Studienarbeit

Backup

Durchschnittliche Fehlerwahrscheinlichkeiten (Pipeline)

Programm	$\overline{R_1^{Bit-Flip}}$	$\overline{R_2^{Bit-Flip}}$	$\overline{R_3^{Bit-Flip}}$	$\overline{R_4^{Bit-Flip}}$
Fibo	0,364	0,373	0,166	0,368
PZ	0,221	0,296	0,232	0,294
QS	0,283	0,362	0,286	0,356

Backup

Durchschnittliche Fehlerwahrscheinlichkeiten (Registerbank)

Programm (Anzahl benutzter Flip-Flops)	$\overline{R_1^{Bit-Flip}}$	$\overline{R_2^{Bit-Flip}}$	$\overline{R_3^{Bit-Flip}}$	$\overline{R_4^{Bit-Flip}}$
Fibo (192 von 992)	0,101	0,110	0,080	0,086
PZ (352 von 992)	0,180	0,224	0,174	0,208
QS (544 von 992)	0,163	0,198	0,148	0,184