TECHNISCHE UNIVERSITÄT DRESDEN

Belegverteidigung

Platzierung und Verdrahtung massiv-paralleler FPGA-Designs am Beispiel eines Many-Core-Prozessors

Michael Lange

Dresden, 03.11.2015

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Gliederung

1 Aufgabenstellung
2 Voraussetzungen und Erweiterungen
3 Algorithmen und Implementierung
4 Maximal mögliche Taktfrequenz
5 Verbesserung der Taktfrequenz
6 Fazit
7 Literatur

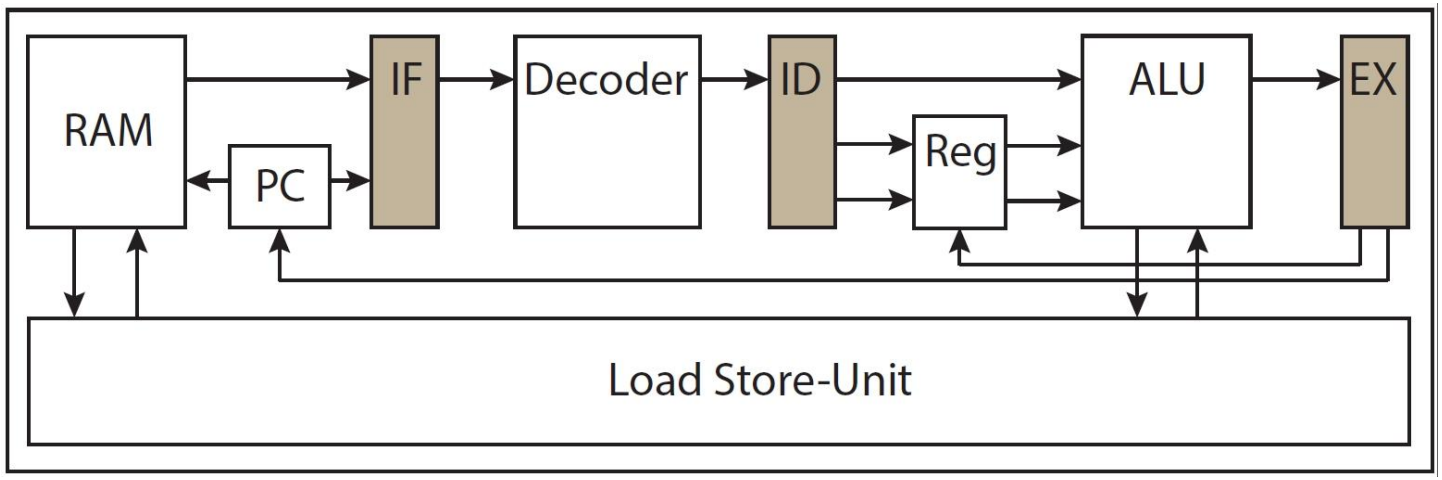
1 Aufgabenstellung

1. Literaturstudium zur geführten Platzierung und Verdrahtung auf FPGAs
2. Auswahl und Analyse von 3 parallelen skalierbaren Algorithmen, die verschiedene Topologien erfordern
3. Simulation, Implementierung und Test auf einer FPGA samt UART
4. Ermittlung der maximalen Taktfrequenz
5. Untersuchung der Möglichkeiten diese zu erhöhen
6. Bewertung der erzielten Skalierbarkeit, Zusammenfassung und Dokumentation der Ergebnisse

2 Voraussetzungen

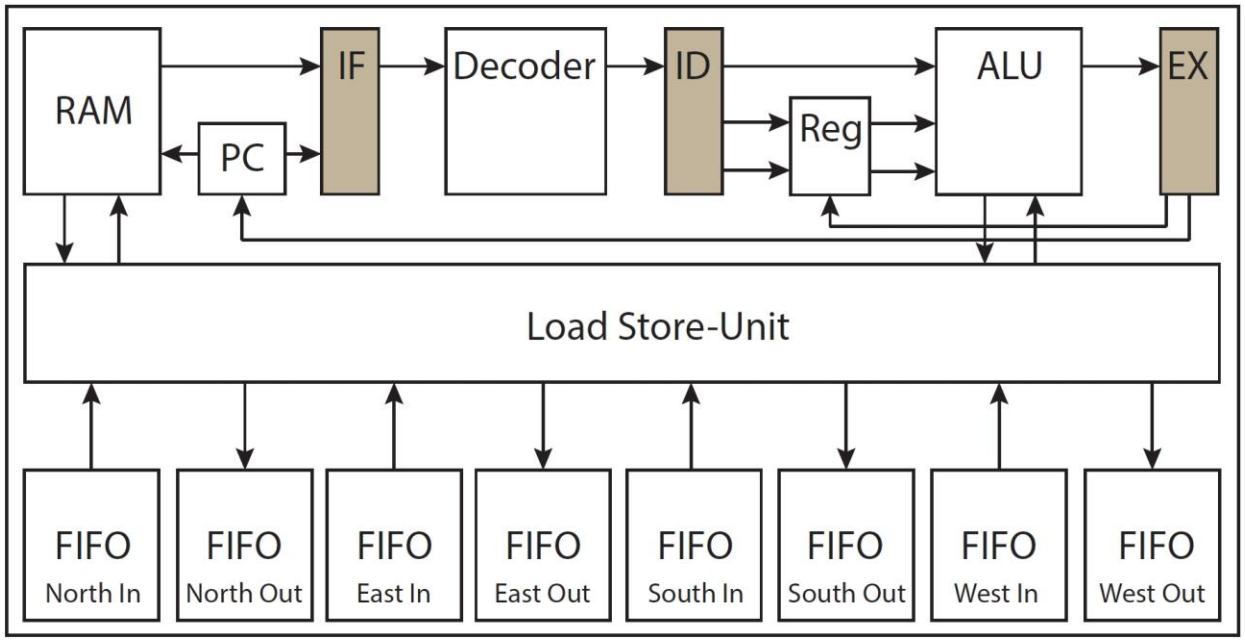
eigenentwickelter MIPS-Kern auf Basis des MIPS R2000 in VHDL aus dem Komplexpraktikum:

- MIPS-Subset (RISC, 32 Bit, ohne Multiplikation und Division)
- 3-stufige Pipeline und Dual-Port-RAM
- GCC-Compiler mit MIPS-Backend, um C-Code in MIPS-Befehle zu übersetzen
- Maximaltakt auf dem Virtex-7: 133 MHz



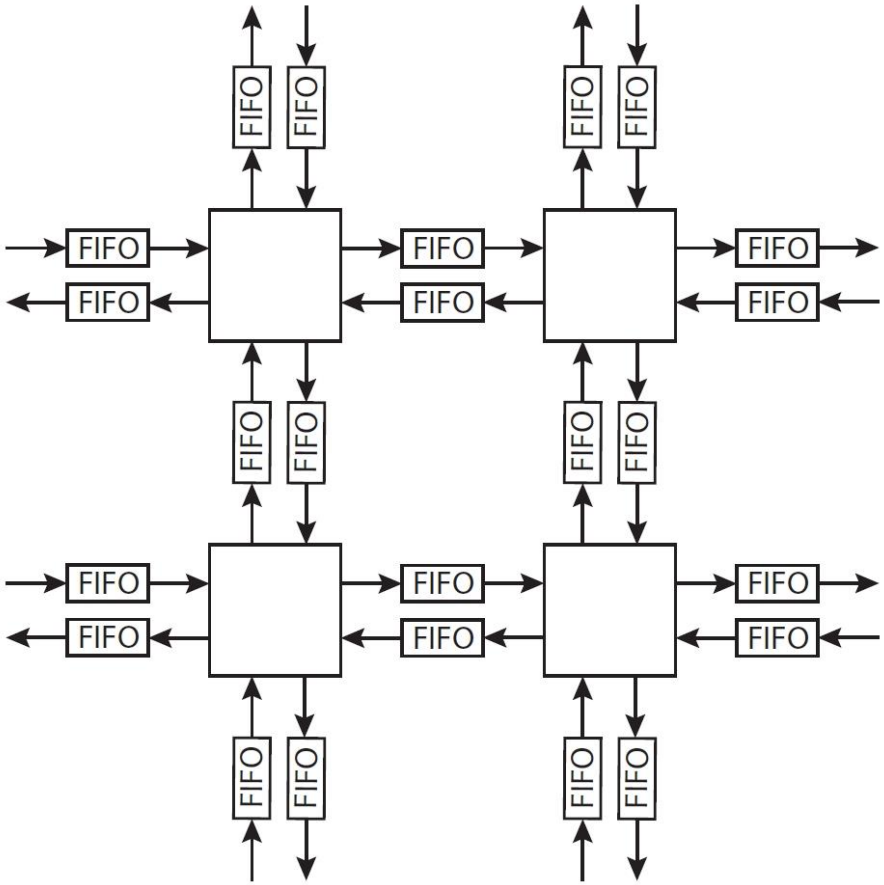
2 Erweiterung des Prozessors

- Hinzufügen von 4 Input- und 4 Output-FIFOs an die Load Store-Einheit
- Memory Mapped I/O für die Core-Nummer und die Status- und Datensignale der FIFOs



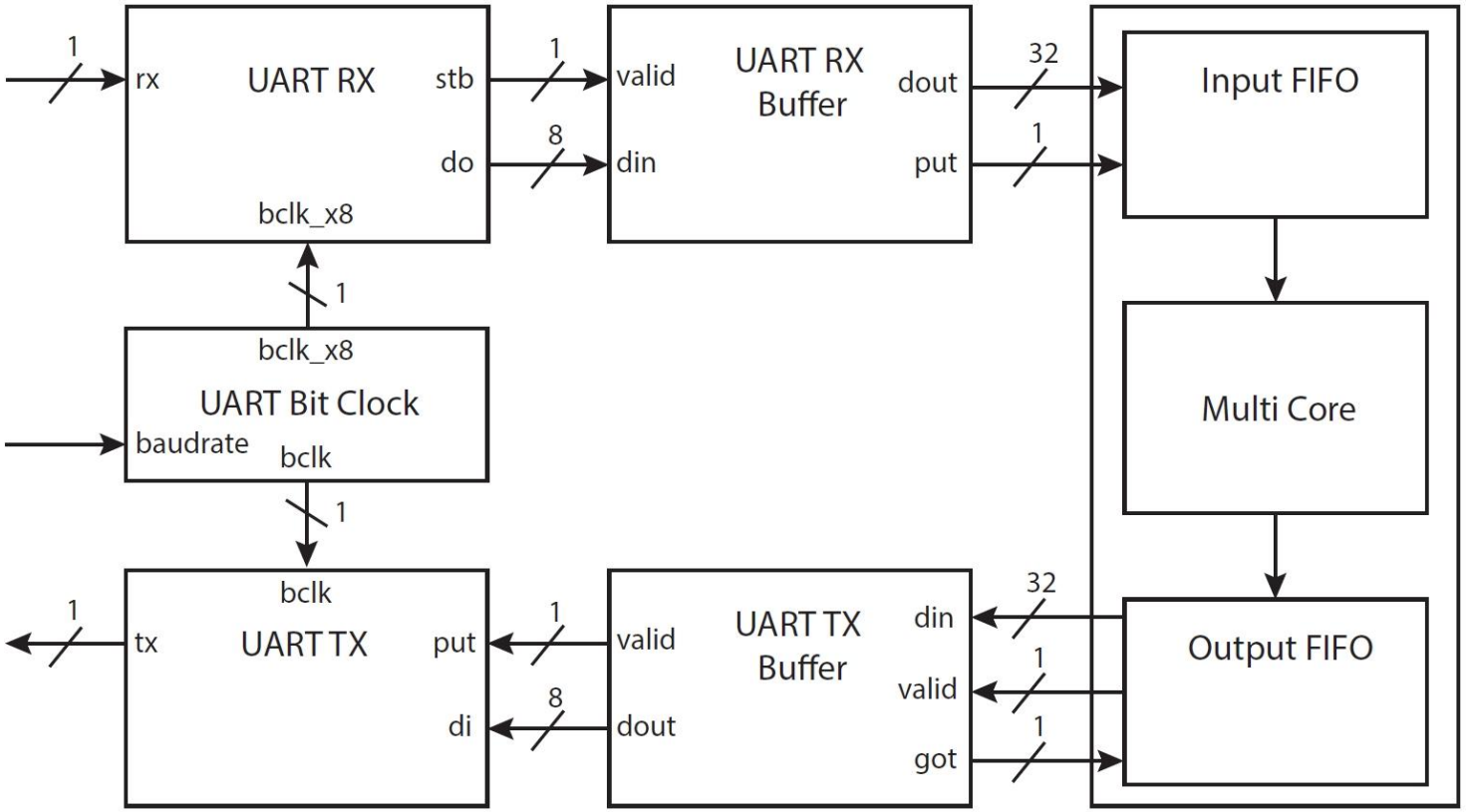
2 Verdrahtung zum Multi Core - Beispiel

- Verbindung der Nachbar-Cores mittels FIFOs
- FIFO-Tiefe variabel
- vor Lesen/Schreiben Überprüfung des FIFO-Status notwendig
- Programmspeicher bei allen Prozessoren gleich
- Jeder Core erhält für ihn spezifischen Programmcode (anhand Core-Nummer und 2. Parameter) durch Case-Anweisung im C-Code



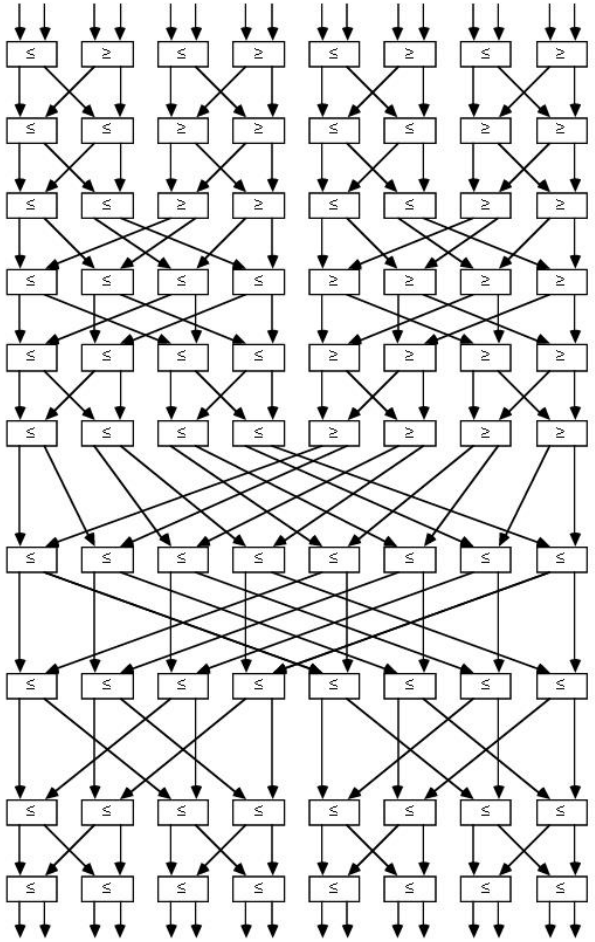
2 Verbindung mit UART

- serielle bitweise Daten-Ein- und –Ausgabe per UART



3 Algorithmen – Bitonischer Sortierer

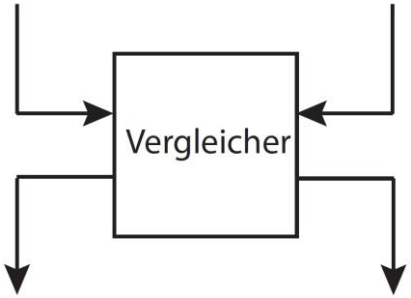
- Folge von Zahlen wird systematisch in immer größere bitonische Folgen sortiert
- Parallele Ausführung in jeder Stufe
- Folge heißt bitonisch, wenn der erste Teil der Folge aufsteigend und der zweite Teil absteigend sortiert (oder die Folge so rotiert werden kann, dass diese bitonisch wird)

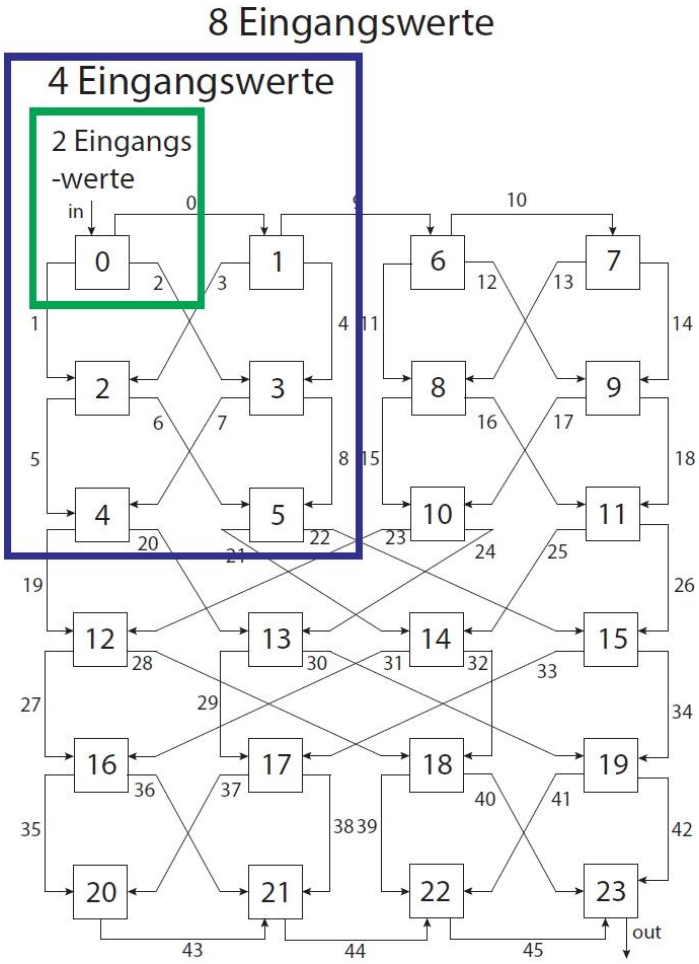


3 Implementation – Bitonischer Sortierer

- Erstellung einheitlicher Vergleicher um Skalierbarkeit zu vereinfachen
- serielle Ein- und Ausgangsstufe benötigt individuelles Design

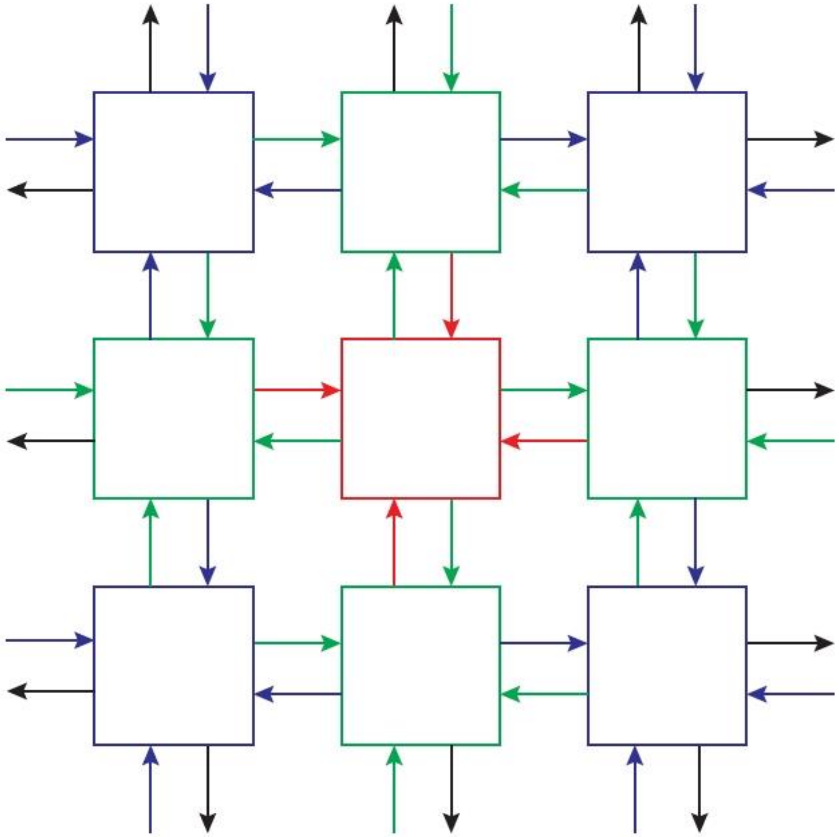
- Skalierbar bis 16 Eingangswerten (80 Prozessoren)





3 Algorithmen – Stencil-Code

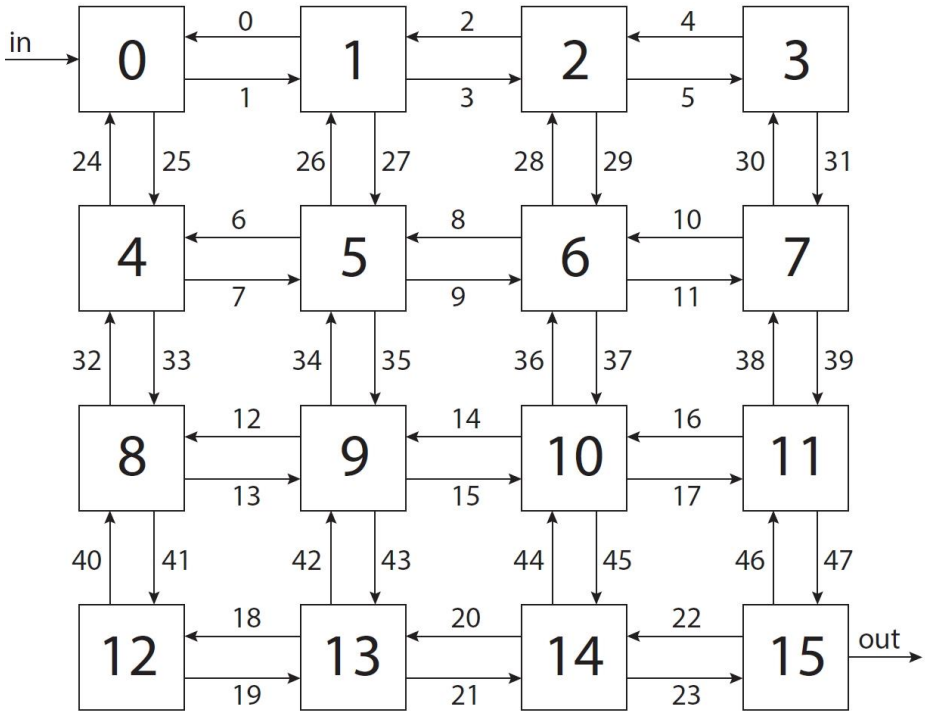
- gewählter Algorithmus:
  - 5-Point 2D Stencil Code
- Berechnungsergebnis eines Kerns von den vier Nachbarn abhängig
- jeder Kern führt selbe Operation aus
- für verschiedenste Algorithmen:
  - partielle Differenzgleichungen
  - lineare Gleichungssysteme
  - Bildverarbeitung
  - Strömungsalgorithmen
  - zellulare Automaten



3 Implementation – Stencil-Code

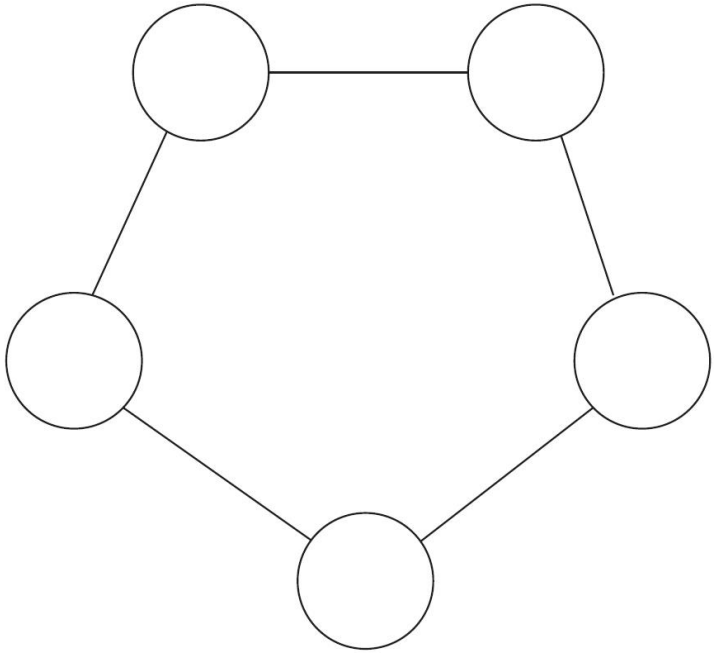
- regelmäßiges, quadratisches Gitter
- 9 verschiedene Himmelsrichtungen möglich
- am Eingang wird konstant ein fester Wert angelegt
- Summe der Nachbardaten dividiert durch 4 (Rechts-Shift)
- Ergebnis wird wiederum an alle Nachbarn geschickt
- Algorithmus endet nicht, es stellt sich aber ein Gleichgewicht ein

- Skalierbar bis zu einer Seitenlänge von 11 (121 Prozessoren)



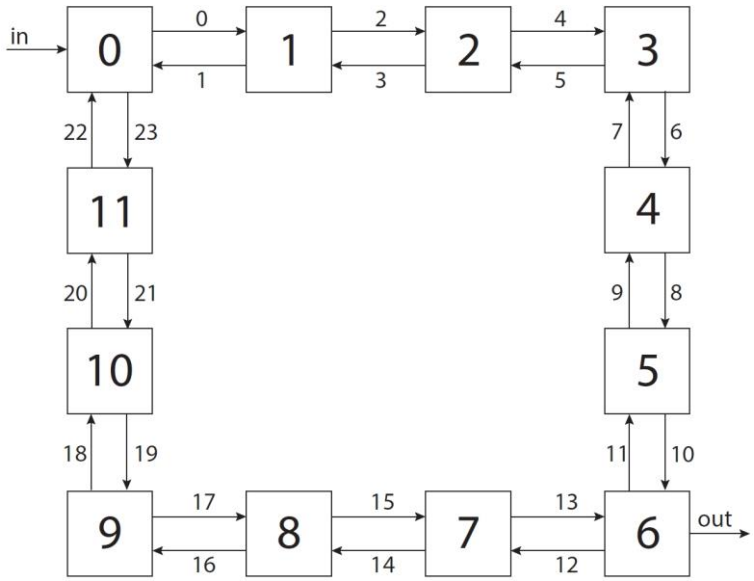
3 Algorithmen – Leader Election

- Netzwerk-Algorithmus mit mehreren Mitgliedern mit genau einem Master
- jeder Teilnehmer sieht nur seine eigene ID
- Datenaustausch zwischen den Mitgliedern
- Ziel: Bestimmung des Teilnehmers mit höchster ID = Master



3 Implementation – Leader Election

- Ring in Form eines quadratischen „Kreises“
- zu Beginn werden die IDs seriell im Uhrzeigersinn zugewiesen
- jeder Teilnehmer ist zu Beginn „wach“
- mittels Verbindungsleitungen werden IDs zu allen Nachbarn transportiert
- wenn beide Nachbar-IDs höherwertiger sind, geht Teilnehmer in „schlafend“ über
- in diesem Zustand schleift er die Nachbar-IDs durch
- Algorithmus läuft solange, bis nur noch der Master „wach“ ist
- dieser schickt letztmalig seine ID in Uhrzeigersinn zum Ausgang



3 Test der Algorithmen

- RTL-Simulation des größten Designs und ausgewählter kleinerer Skalierungsstufen
- Implementierung des größten Designs per Remote-Zugriff auf die FPGA-Cloud auf den Virtex-7
- Erstellung von Stimuli-Eingangsdateien mittels Hex-Editor
    - Stencil-Algorithmus: zufällige Eingabewerte
    - Bitonischer Sortierer: aufsteigende, absteigende und zufällig sortierte Folgen
    - Leader Election: Worst-Case, Best-Case und zufällige Werte
- Anlegen der Input-Dateien per Minicom an den UART-Eingang
- Schreiben der Ausgabewerte in eine Output-Datei
- Überprüfung auf Korrektheit der ausgegebenen Werte

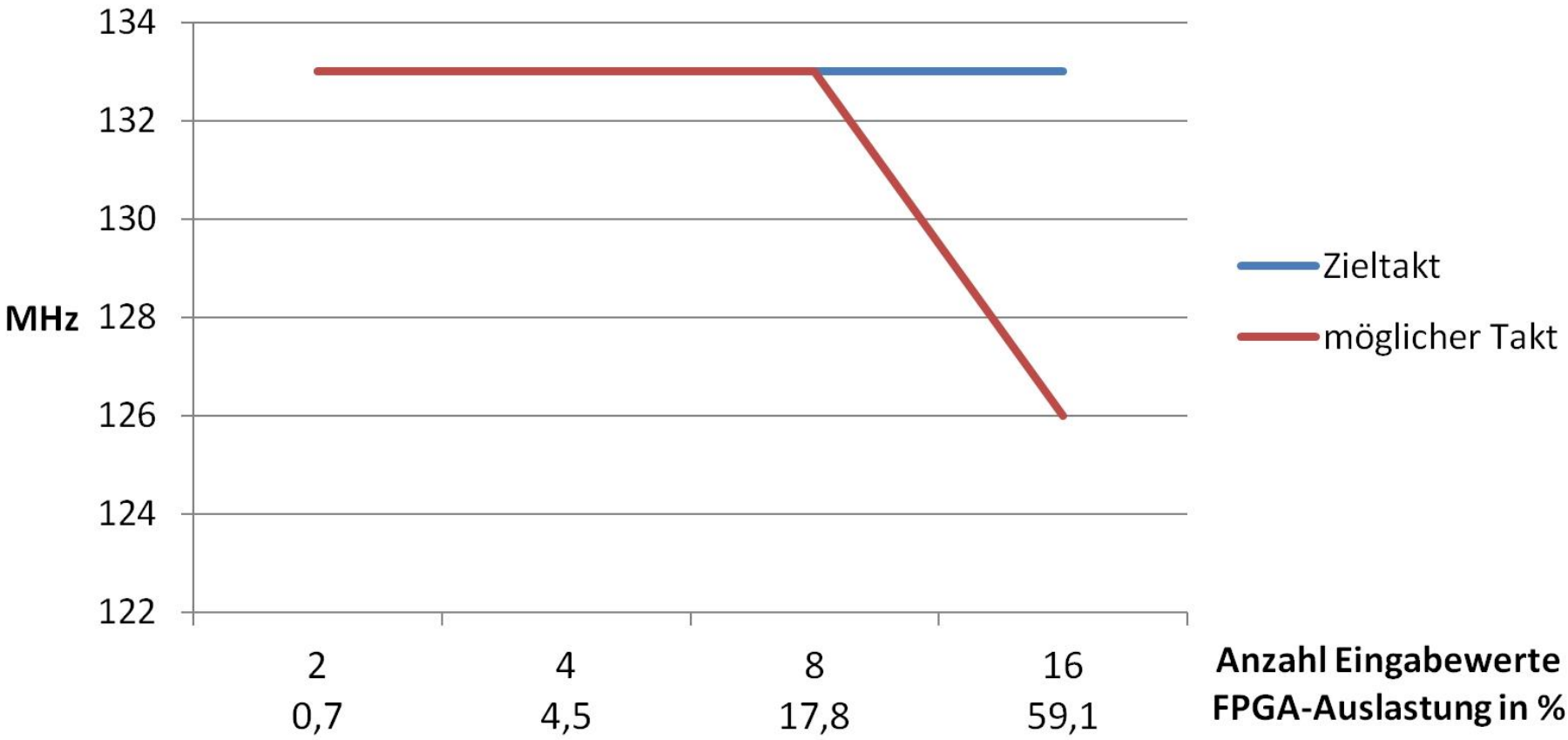
4. Ermittlung der maximalen Taktfrequenz

- Mittels Over-Constraining, nach P&R ermittelt
  -> Taktfrequenz auf 133 MHz (Zieltakt) eingestellt
  -> Ermittlung der max. Taktfrequenz anhand des negativen Slacks

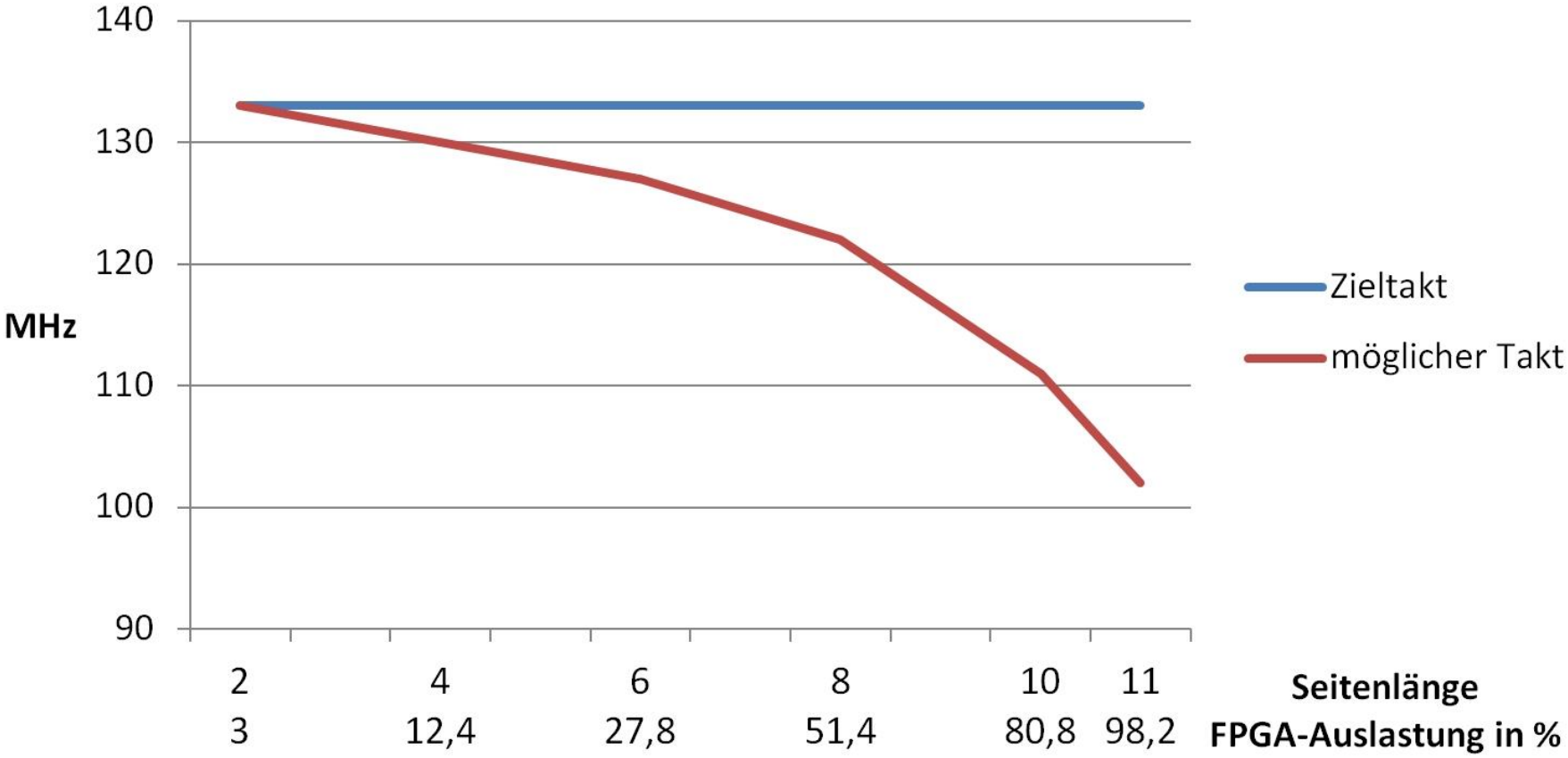
$$f_{max} = \frac{1}{7{,}5ns - wns}$$

- nur bedingt genau, da Tool bei Nichterreichung abbricht, bedingt durch die Heuristik aber auch etwas höhere Frequenzen möglich wären
- Ermittlung der Frequenz über gewählte Skalierungsstufen der Algorithmen

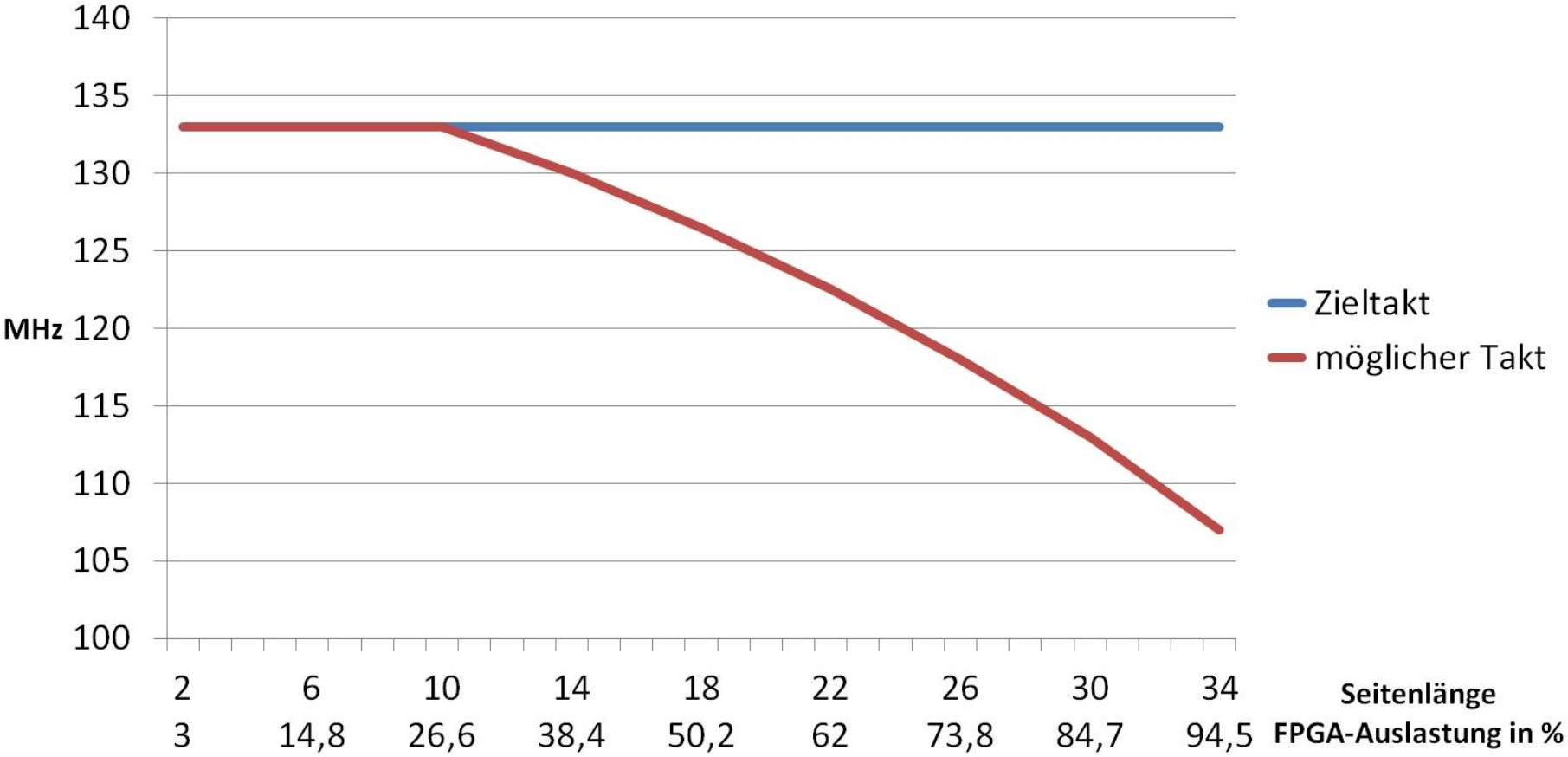
4. Ermittlung der maximalen Taktfrequenz - Bitonic



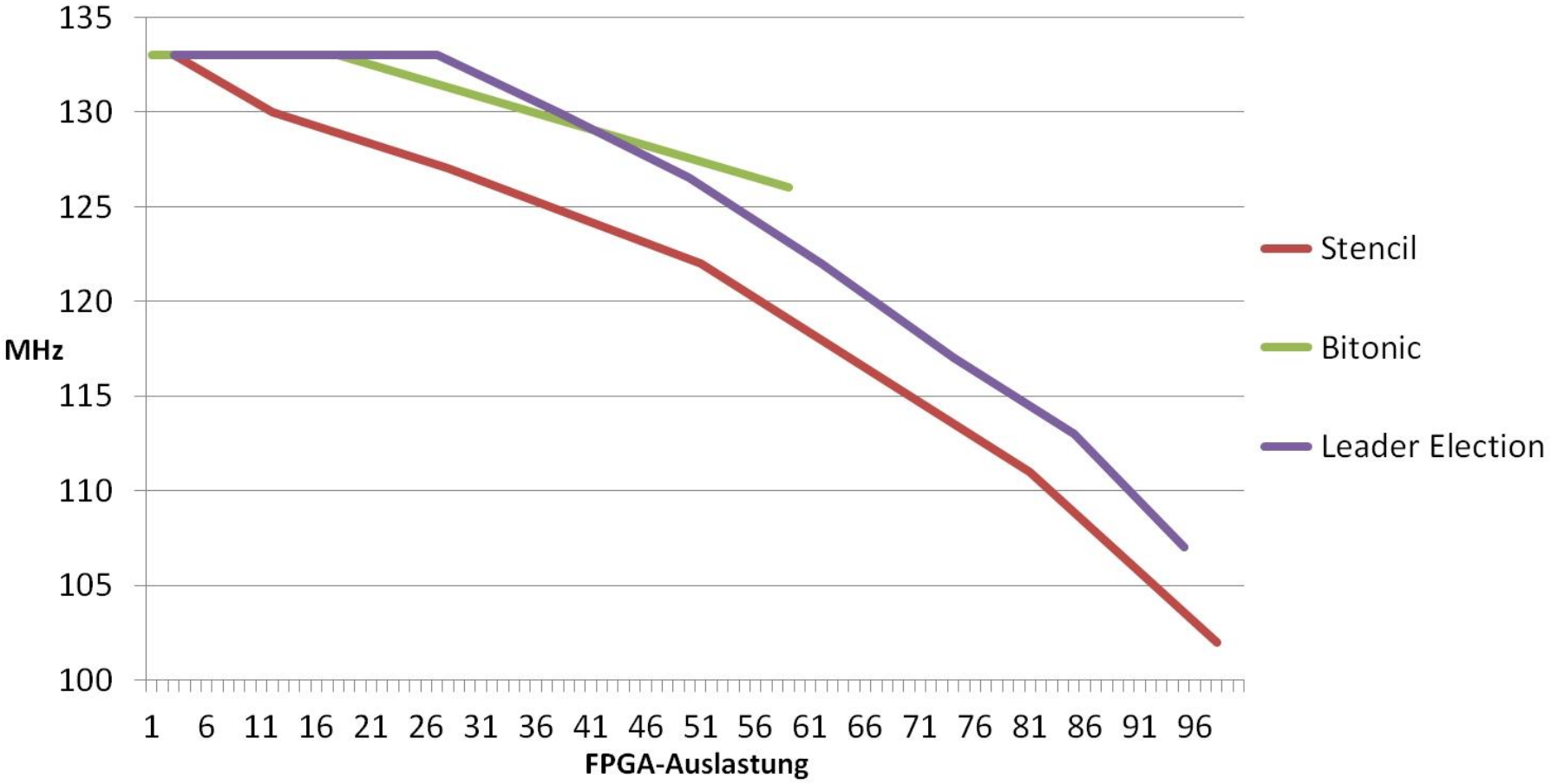
4. Ermittlung der maximalen Taktfrequenz - Stencil



5. Ermittlung der maximalen Taktfrequenz – Leader Election

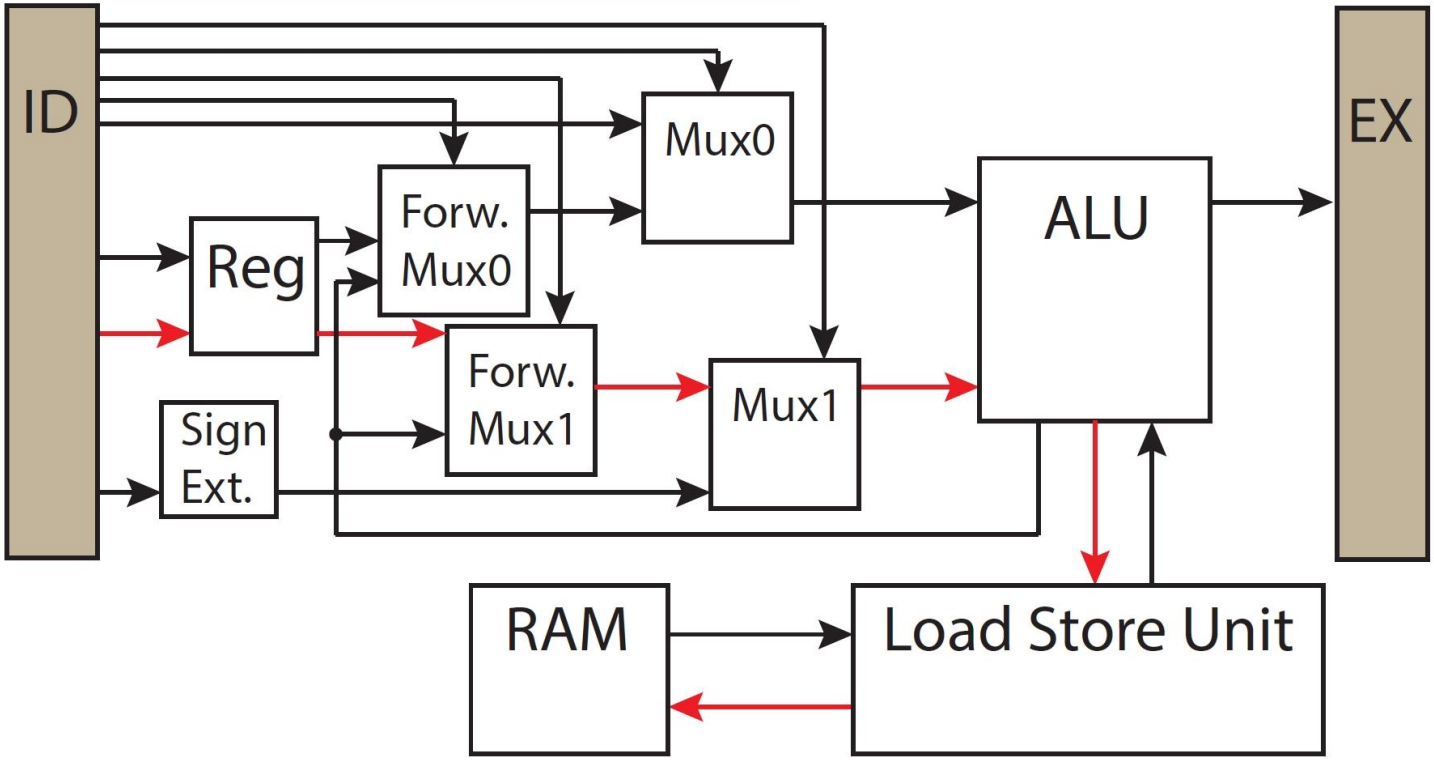


4. Ermittlung der maximalen Taktfrequenz – Zusammenfassung



5. Verbesserung der maximalen Taktrate

-> Verringerung der Länge des kritischen Pfades



5. Verbesserung der maximalen Taktrate

mittels ***User-Constraints:***

- Platzierung einzelner LUTs, FFs, BRAMs, …
- Platzierung einzelner Baublöcke
- Fanout begrenzen (LUT- und Register-Duplizierung)
- einzelne Pfad-Laufzeit festlegen
- In- und Output-Delays festlegen
- PINs festlegen

Fazit:

- keine Verbesserung des Timings bei großen Designs
- auch kleinste Änderungen wirken sich negativ aus
- durch Placement-Contraints konnte Place&Route-Zeit reduziert werden

5. Verbesserung der maximalen Taktrate

mittels ***Synthese- und Implementations-Strategien*** in Vivado

- vordefinierte tcl-Constraint-Scripts
- zur Optimierung folgender Eigenschaften:

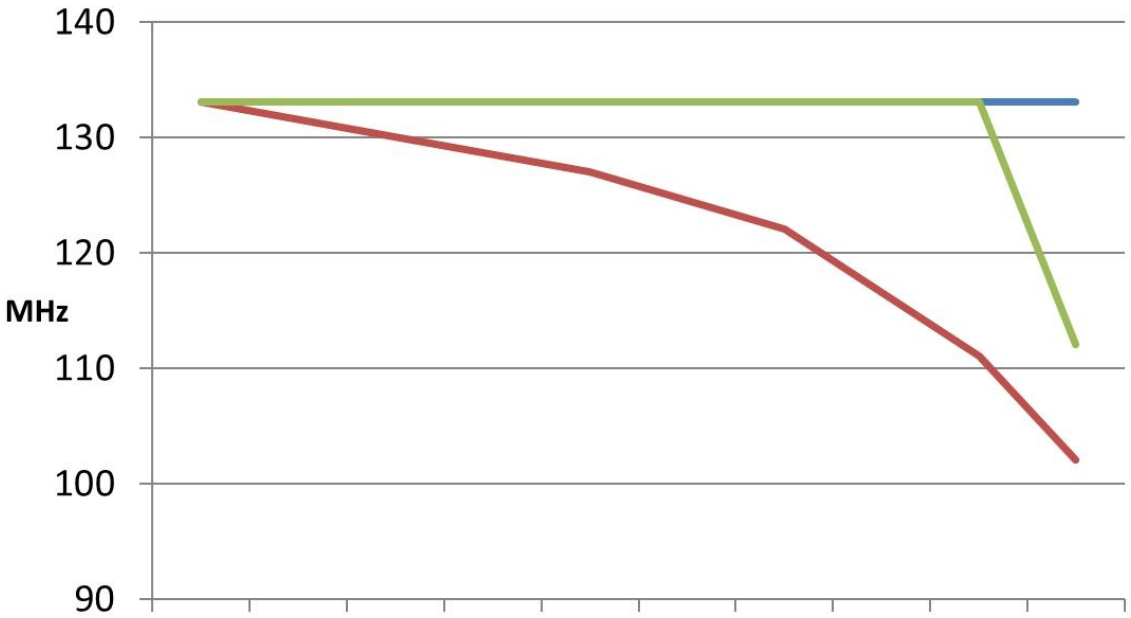
  => Performance, Platzbedarf oder Verlustleistung

Getestete Strategie: Performance bzw. Frequenz-Optimierung

- setzen eines globalen Fan-Outs
- Aktivierung von LUT- und Register-Duplikationen
- FSM-Kodierung in One-Hot
- Gewählte Strategien:
  - Synthese: *Flow_PerfOptimized_High*
  - Implementation: *Performance_Explore*

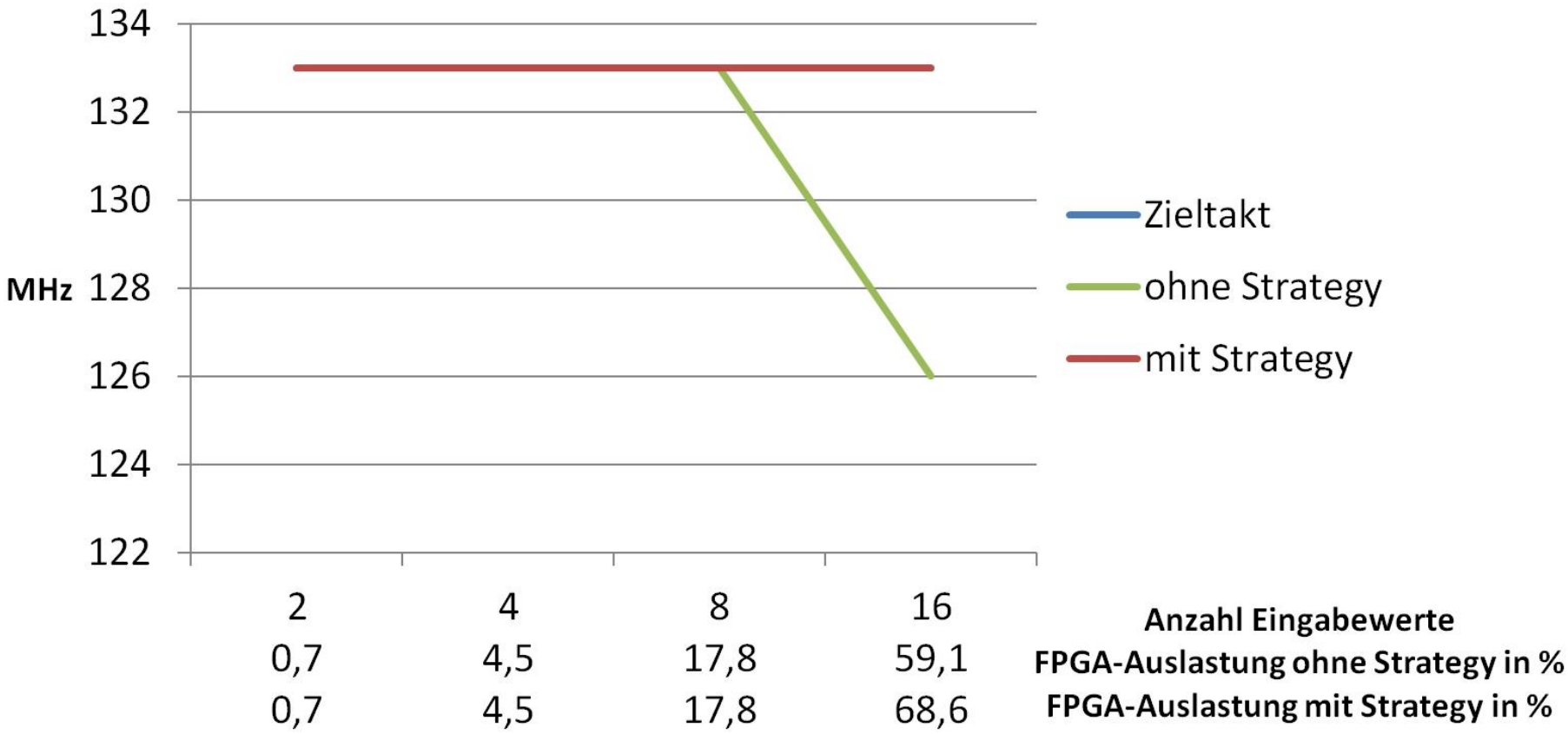
- auch Erstellung eigener Strategien per GUI möglich

5. Benutzung der Strategien - Stencil



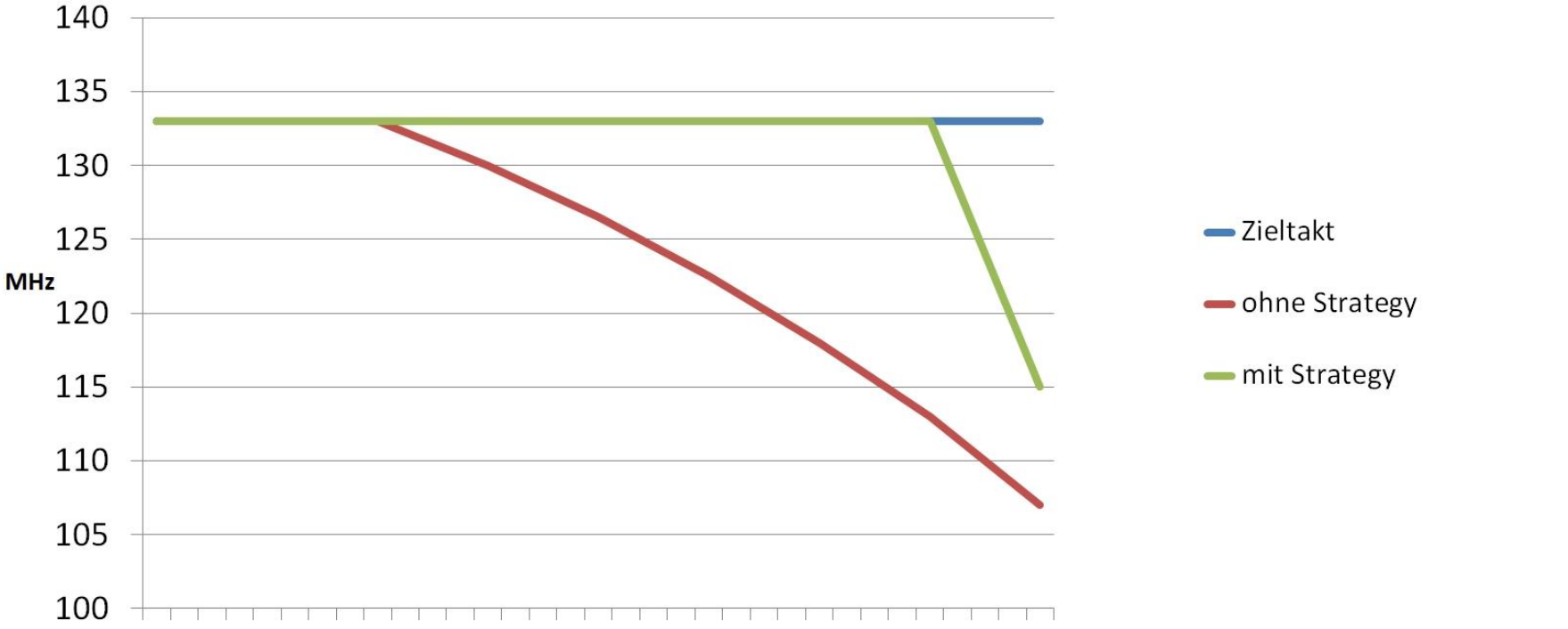
Seitenlänge	2	4	6	8	10	11
FPGA-Auslastung ohne Strategy in %	3	12,4	27,8	51,4	80,8	98,2
FPGA-Auslastung mit Strategy in %	3	13,8	31,4	56,1	88	99,6

5. Benutzung der Strategien - Bitonic



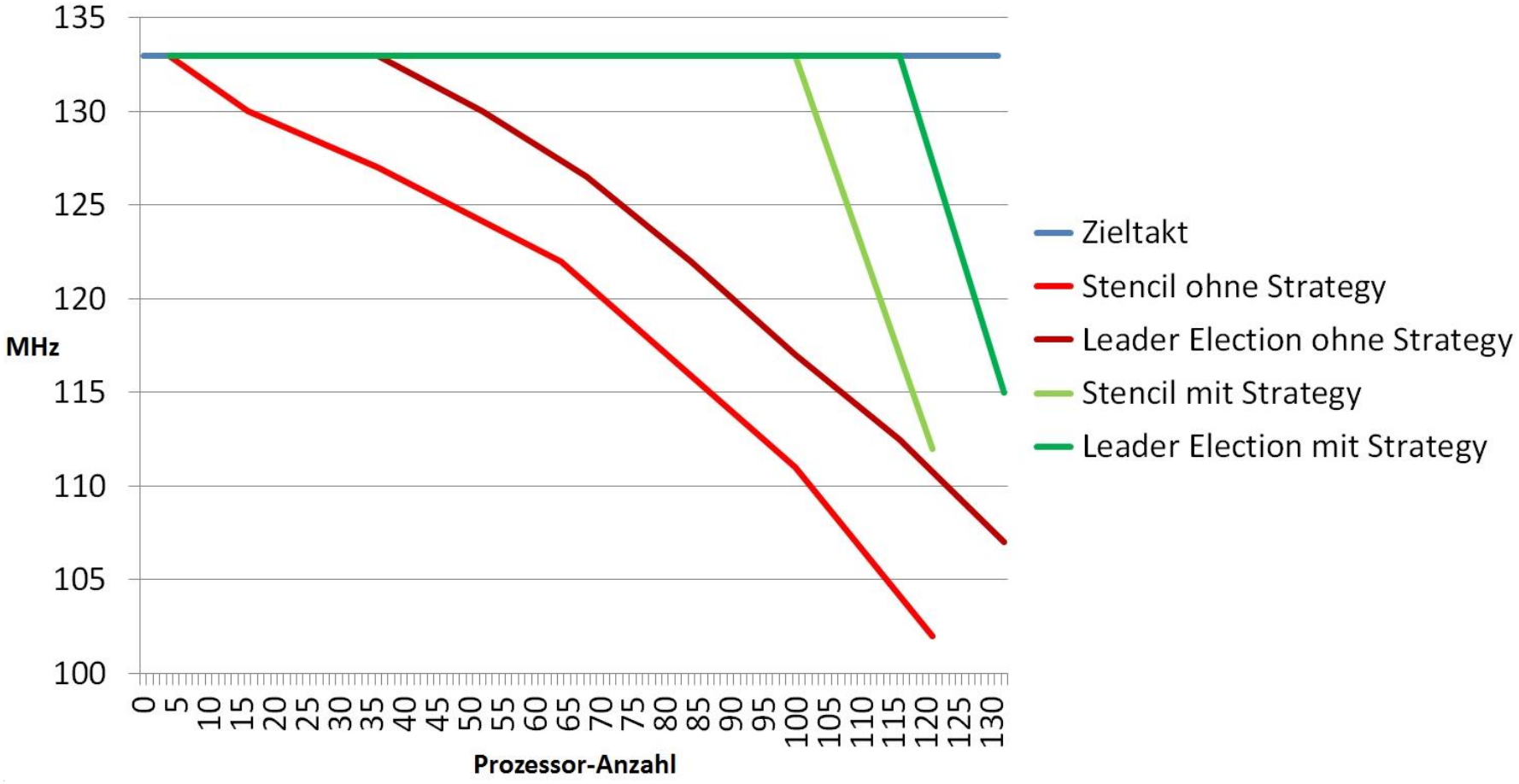
Anzahl Eingabewerte	2	4	8	16
FPGA-Auslastung ohne Strategy in %	0,7	4,5	17,8	59,1
FPGA-Auslastung mit Strategy in %	0,7	4,5	17,8	68,6

5. Benutzung der Strategien – Leader Election



Seitenlänge	2	6	10	14	18	22	26	30	34
FPGA-Auslastung ohne Strategy in %	3	14,8	26,6	38,4	50,2	62	73,8	84,7	94,5
FPGA-Auslastung mit Strategy in %	3	14,8	26,6	41,7	58,2	73,1	86,7	96	98,9

5. Benutzung der Strategien – Zusammenfassung



6 Fazit

- keine Verbesserung mit manuellen User-Constraints erzielt
- Strategien bzw. tcl-Scripts mit globalen Änderungen eher geeignet um große Design performanter zu gestalten
- hierfür werden LUTs und Register dupliziert
- Zieltakt erreichbar sofern genügend ungenutzte Ressourcen vorhanden
- beim selbst-erstellten MIPS-Design maximal 18% mehr LUTS und 1,7% mehr Flipflops und im Schnitt 12% mehr LUTs und 1,2% mehr Flipflops nötig um Zieltakt zu erreichen

7 Literatur

Stefan Lankes - Revisiting Shared Virtual Memory Systems for Non-Coherent Memory-Coupled Cores
International Workshop on Programming Models and Applications for Multicores and Manycores - 2012

Larry McMurchie - Placement and Routing for FPGAs
Very Large Scale Integration Systems (VLSI), IEEE Transaction Dezember 1995

Marco Aiello und Eirini Kaldeli - Leader Election in rings
University of Groningen - 2009

Dr. Andreas Schwil - Bitonisches Sortieren
Uni Potsdam - 1999

Parallel Sorting Algorithms
Massey University New Zealand - 2009

Yongpeng Zhang und Frank Mueller - Auto-Generation and Auto-Tuning of 3D Stencil Codes on GPU Clusters
Proceedings of the Tenth International Symposium on Code Generation and Optimization 2012

Xilinx - 7 Series FPGAs Congurable Logic Block. Documentation - 2014.
Xilinx - UltraFast Design Methodology Guide for the Vivado Design Suite v2015.1 - 2015