Vortrag zum Diplom

Untersuchungen zur effizienten Implementierung eines mathematischen Algorithmus in einem FPGA am Beispiel eines Sudoku-Lösers

Michael Dittrich, michael-dittrich@mailbox.tu-dresden.de

Dresden, 04.11.2008

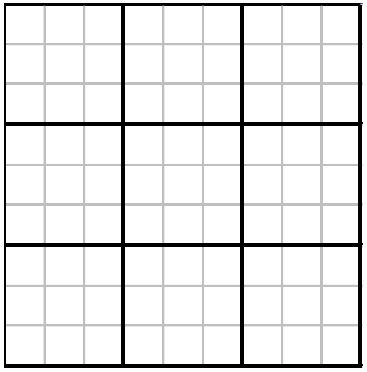
Gliederung

1. **Motivation**
2. **Constraint Satisfaction Probleme**
3. **Exact Cover Probleme**
4. **Hardwareimplementierung**
5. **Messung**
6. **To Do**

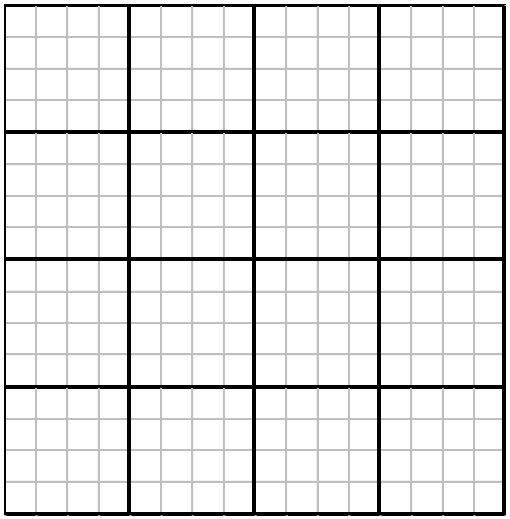
1. Motivation

Wozu die Implementierung der Algorithmen auf FPGA

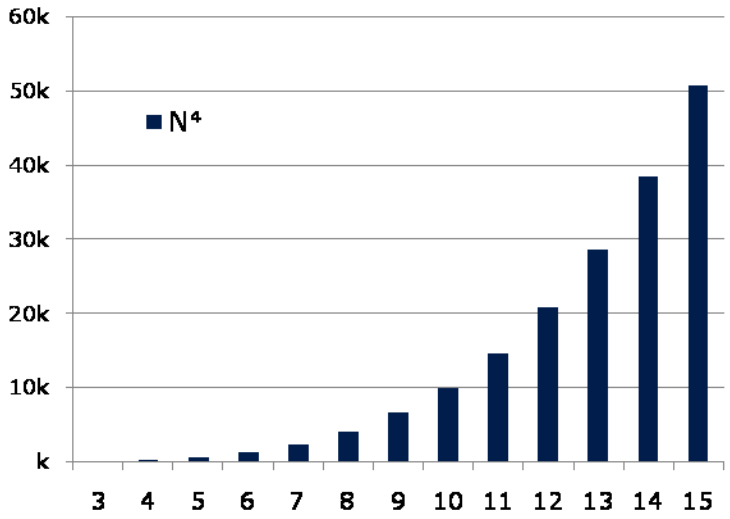
- **(hohe) Leistungsgewinne gegenüber Std-Prozessoren**
  - mehr Performance pro Watt
  - Lösung komplexer mathematischer Probleme (diskrete Optimierung)



N = 3, N^4 = 81



N = 4, N^4 = 256



2. Constraint Satisfaction Problem

Definition

- **Spezifikation eines Problems anhand von Bedingungen und Variablen**
  - Lösung des Problems ohne Angabe eines Lösungsalgorithmus!

$$
\begin{aligned}
CSP &= (X, D, C) \\
X &= \{x_1, ..., x_n\} \\
D &= \{D_{x1}, ..., D_{xn}\} \\
C &= \{C_1, ..., C_k\} \; mit \; C_i \subseteq X^l \times D^l
\end{aligned}
$$

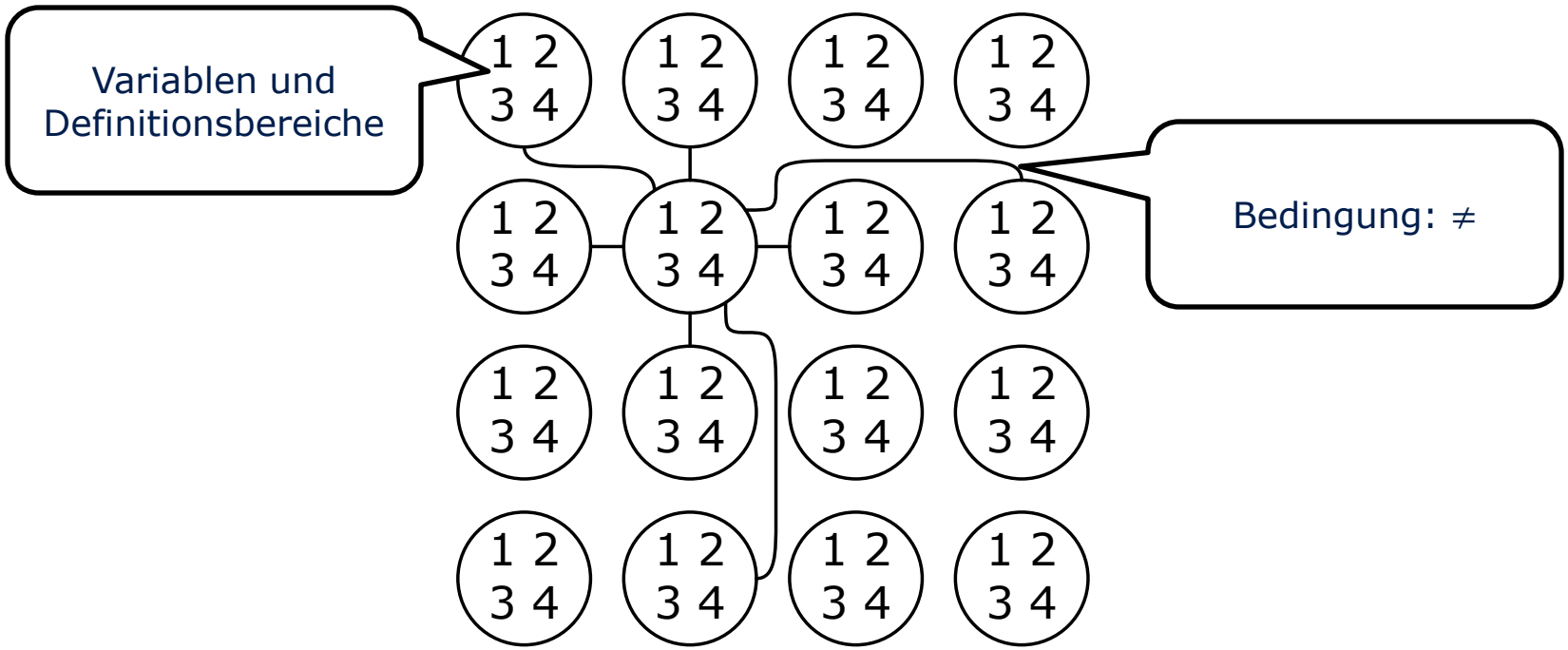
2. Constraint Satisfaction Problem

Sudoku der Ordnung N=3 als CSP

$$
\begin{aligned}
Sudoku &= (X, D, C) \\
X &= \{x_{ij} | i, j \in \{1, ..., 9\}\} \\
D_{x_{ij}} &= \{1, ..., 9\}\ wobei\ i, j \in \{1, ..., 9\} \\
C &= \{\ \forall i \in \{1, ..., 9\}\ alldifferent(x_{i1}, ..., x_{i9}), \\
&\quad\ \forall j \in \{1, ..., 9\}\ alldifferent(x_{1j}, ..., x_{9j}), \\
&\quad\ \forall i, j \in \{z \in \{1, ..., 9\} |\ z\ mod\ 3 = 1\} \\
&\qquad alldifferent(x_{ij}, x_{i(j+1)}, x_{i(j+2)}, x_{(i+1)j}, x_{(i+1)(j+1)}, \\
&\qquad x_{(i+1)(j+2)}, x_{(i+2)j}, x_{(i+2)(j+1)}, x_{(i+2)(j+2)}) \\
&\quad\ \}
\end{aligned}
$$

2. Constraint Satisfaction Problem

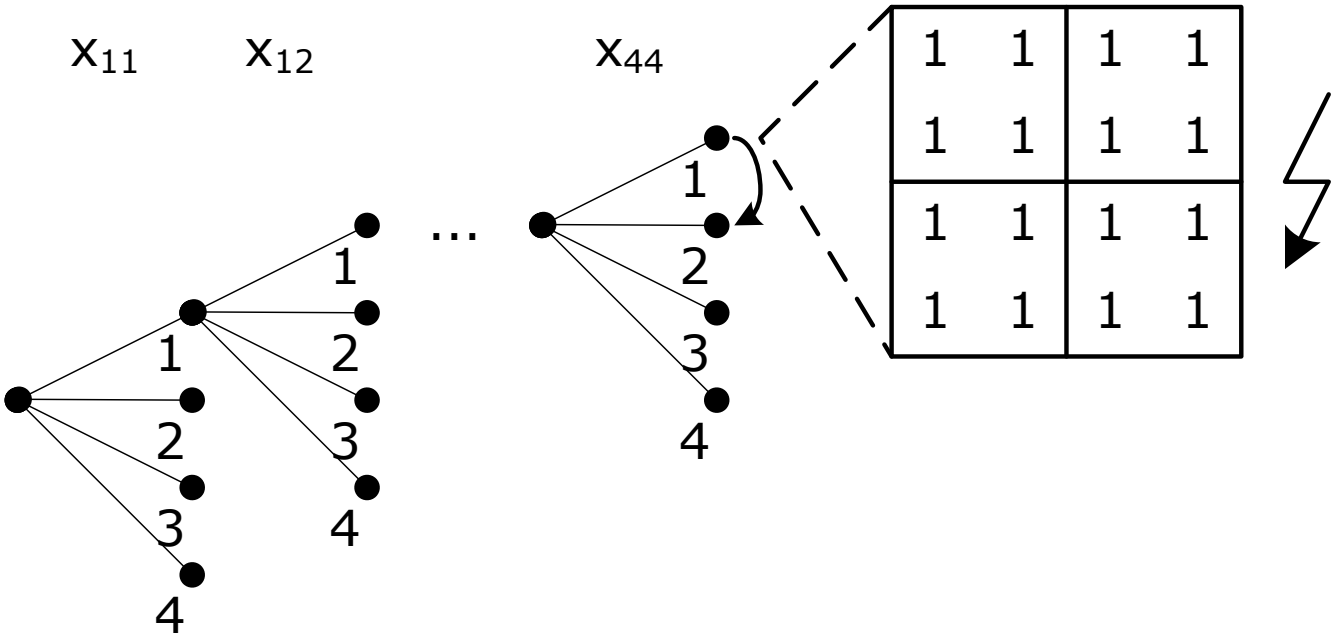
Darstellung als Graph



2. Constraint Satisfaction Problem

Generate & Test

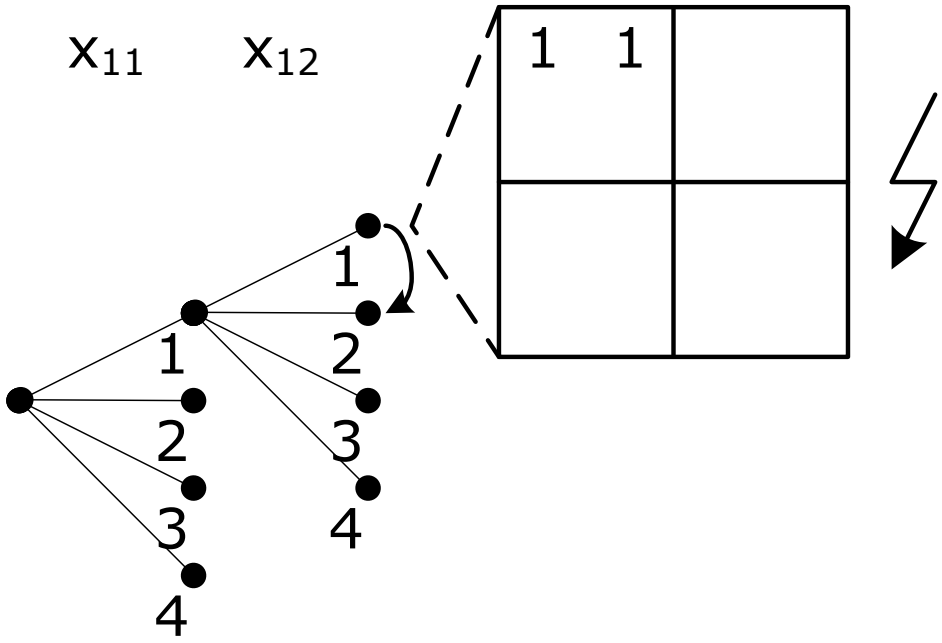
- **Vollständige Wertzuweisung und anschließende Überprüfung der Bedingungen**



2. Constraint Satisfaction Problem

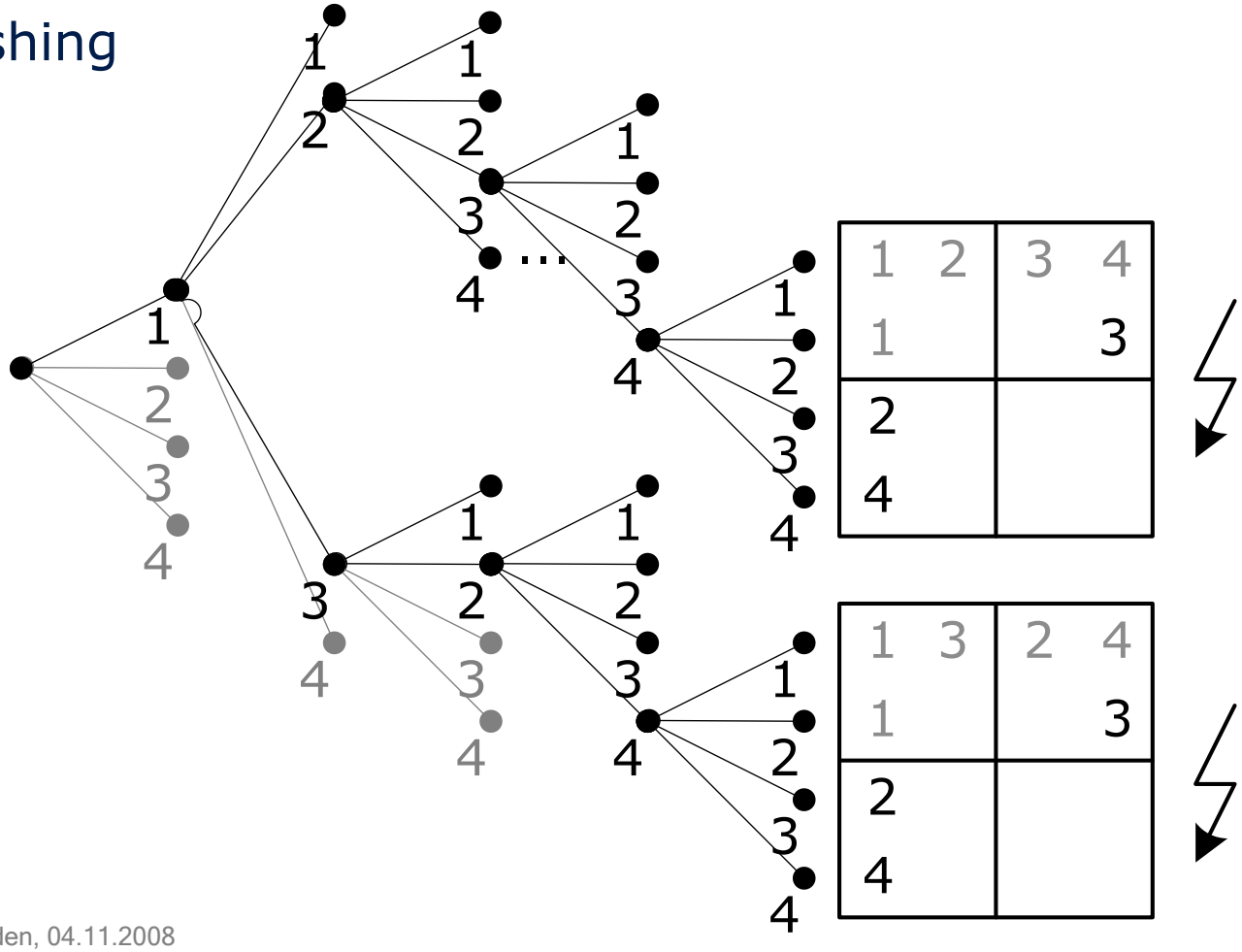
Backtracking

- **Überprüfung der Bedingungen nach jeder einzelnen Wertzuweisung**



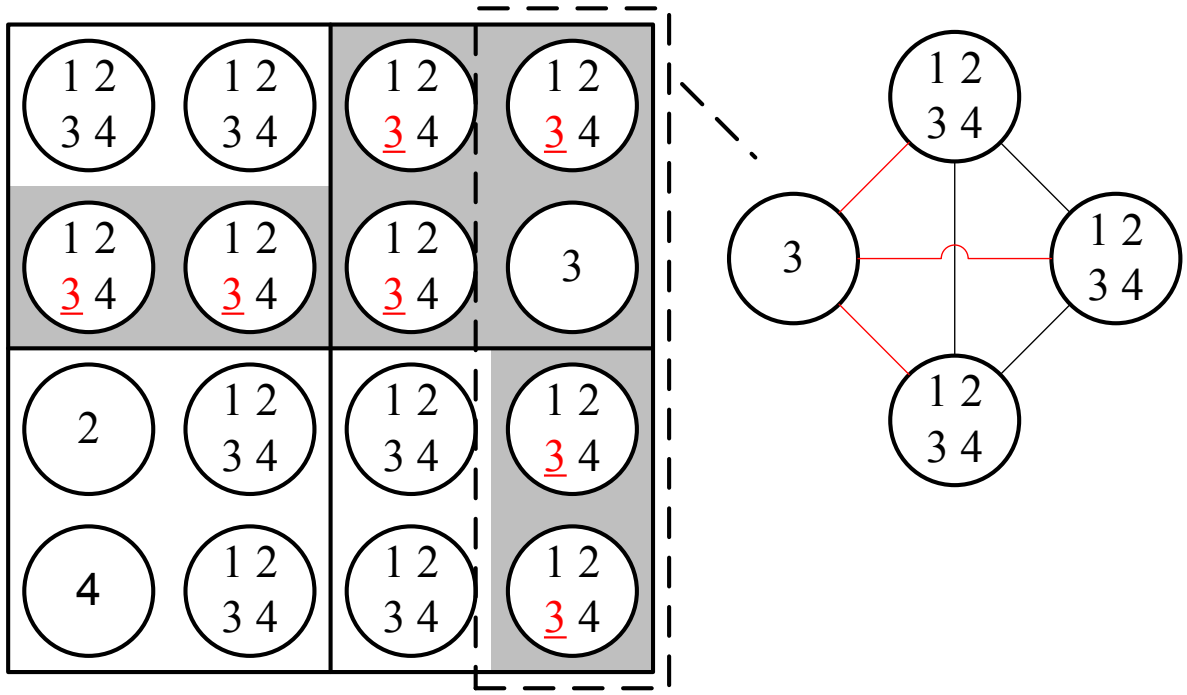
2. CSP

Trashing



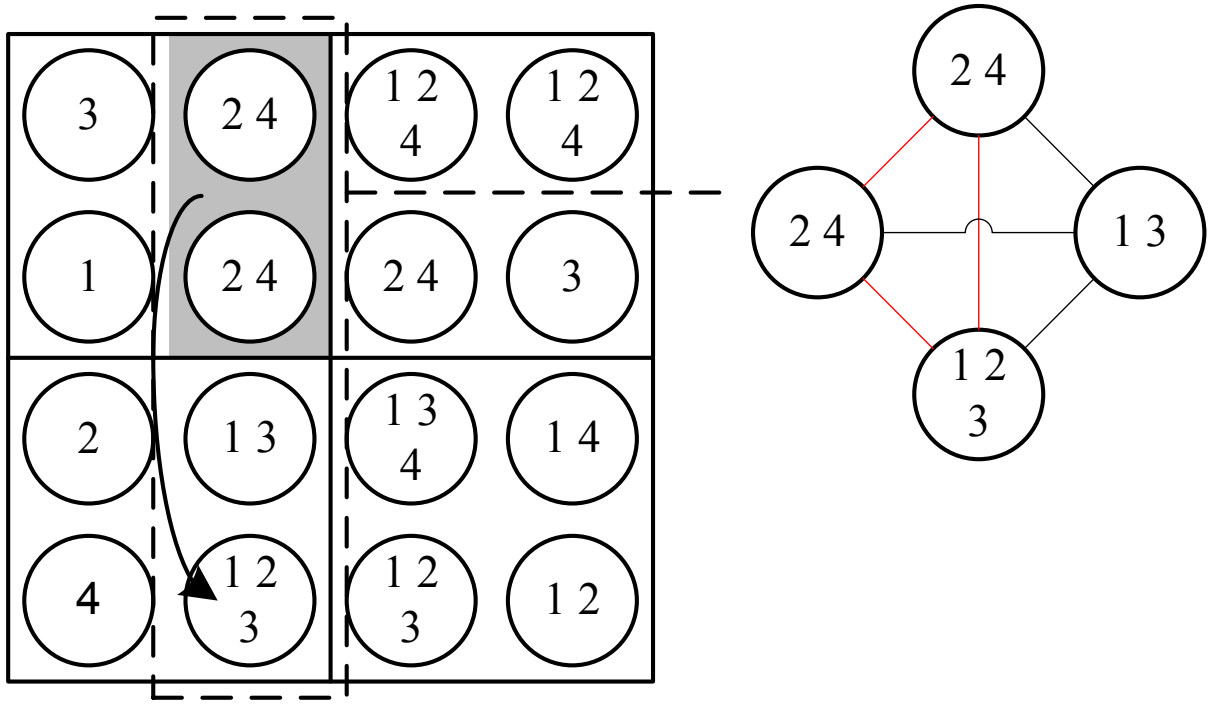
2. Constraint Satisfaction Problem

Node-Consistency, Arc-Consistency



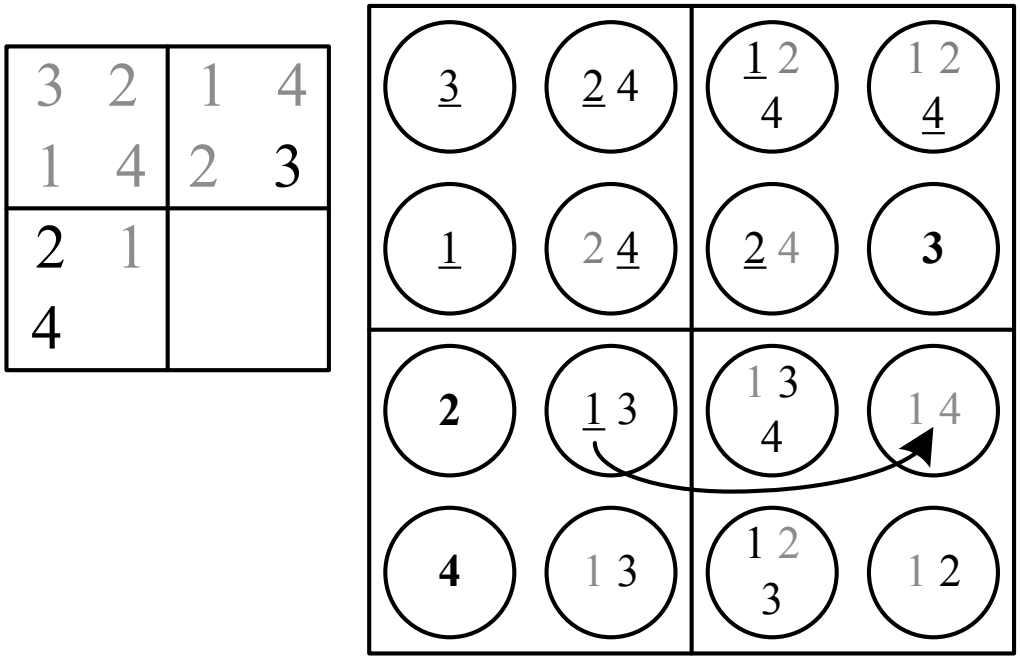
2. Constraint Satisfaction Problem

Path-Consistency



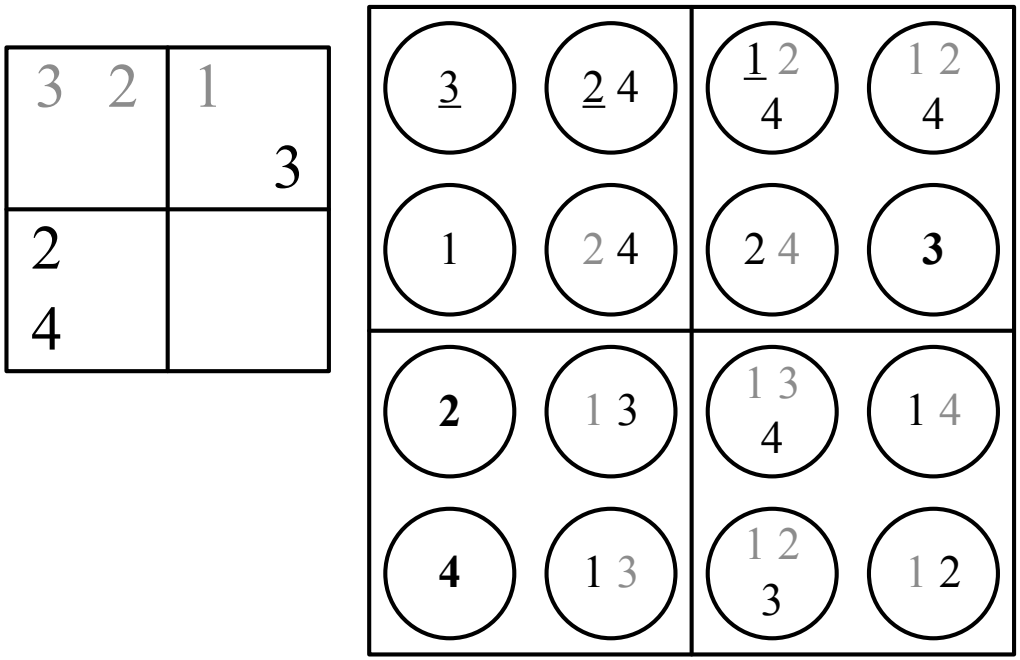
2. Constraint Satisfaction Problem

Forward Checking



2. Constraint Satisfaction Problem

Really Full Lookahead



2. Constraint Satisfaction Problem

Weitere Methoden

- **Search Rearrangement**
  - Instanziierung beginnend bei Variablen mit wenigen verbleibenden Werten
- **Dependency Directed Backtracking**
  - Backtracking direkt zur bedingungsverletzenden Wertzuweisung
- **Nichtdeterministische Algorithmen**
  - Simulated Annealing, etc.

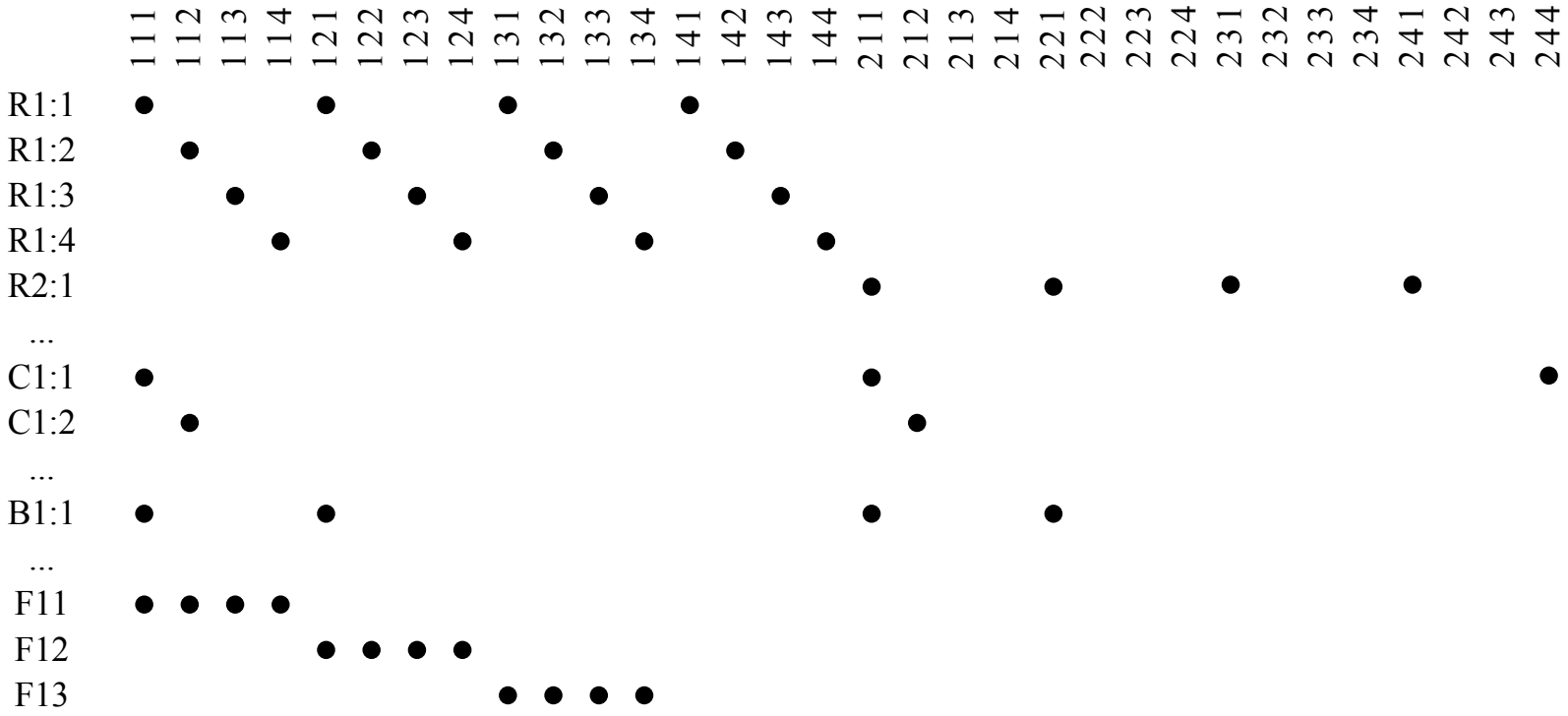
3. Exact Cover Problem

Definition

$$\begin{aligned} X &= \{x_1, ..., x_n\} \\ S &\subseteq P(X) \end{aligned}$$

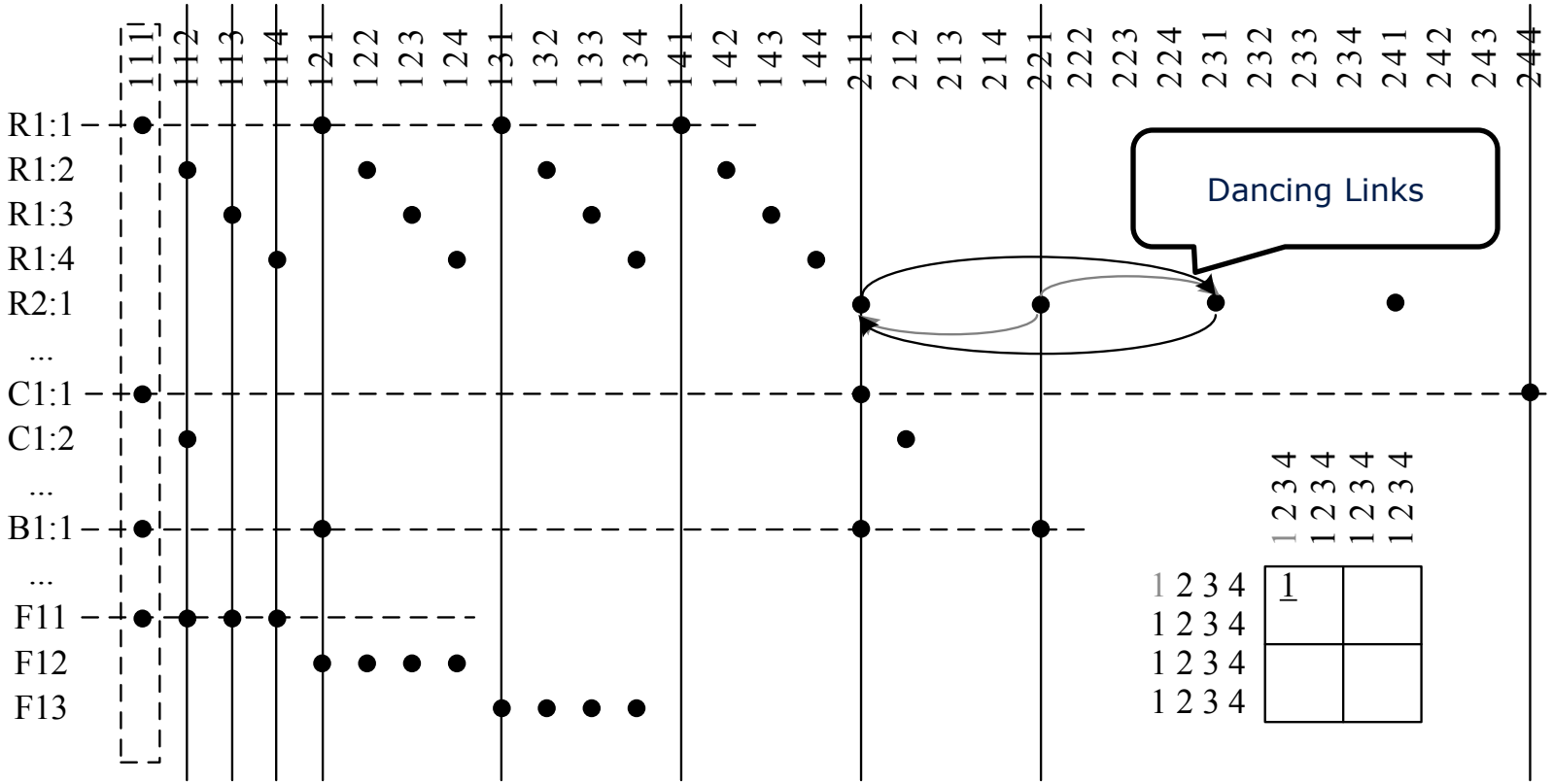
3. Exact Cover Problem

Sudoku als Exact Cover Problem



3. Exact Cover Problem

Lösen des Exact Cover Problems



4. Hardwareimplementierung

Anforderungen

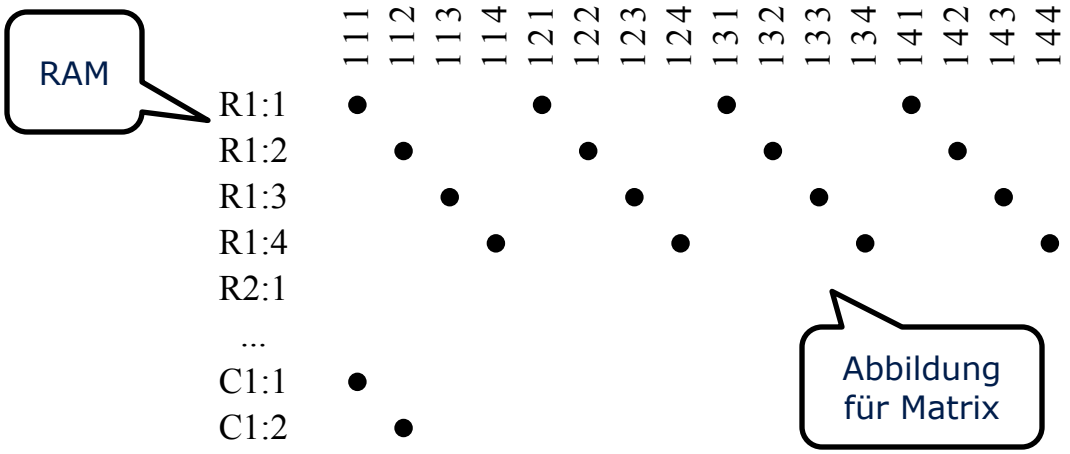
- **Wenig Ram**
  - Sudoku N=15: ≈ 50kB pro Spielfeld → ≈ 2,5 GB für kompletten Backtrack-Stack
  - → Rekonstruieren der alten Spielfelder aus den gesetzten Werten
  - → Datenkomprimierung

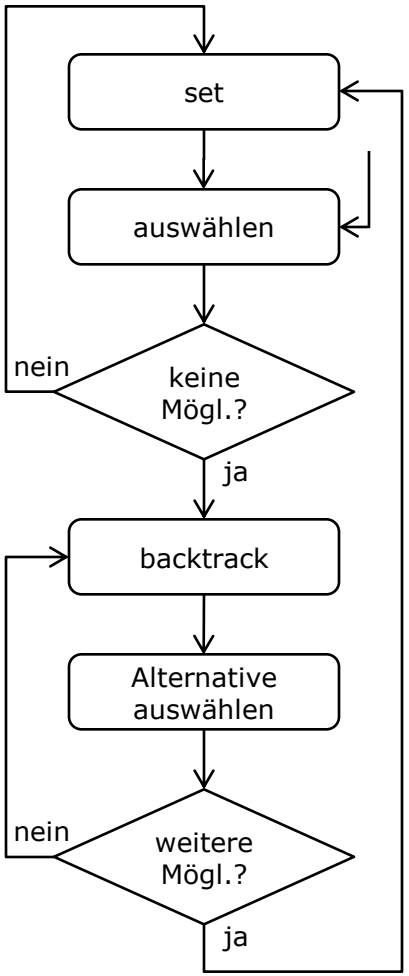
- **Parallelisierung**
  - Partitionierung der Probleme
  - MISD, SIMD

4. Hardwareimplementierung

Algorithmus

- Exact Cover
- Search Rearrangement
- Forward Checking (1/2)
- Dependency Directed Backtracking (1/10)

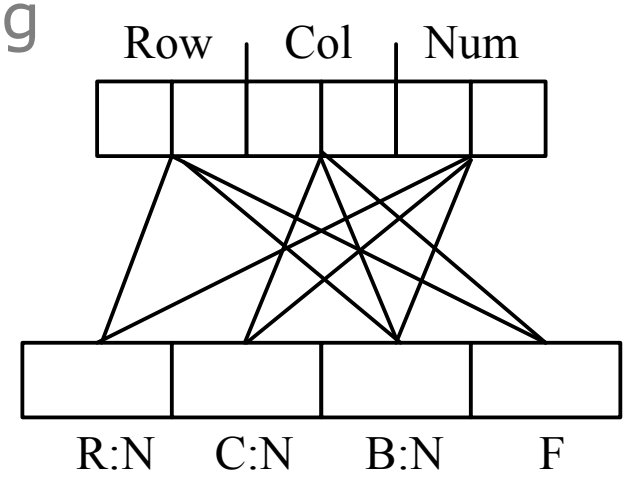


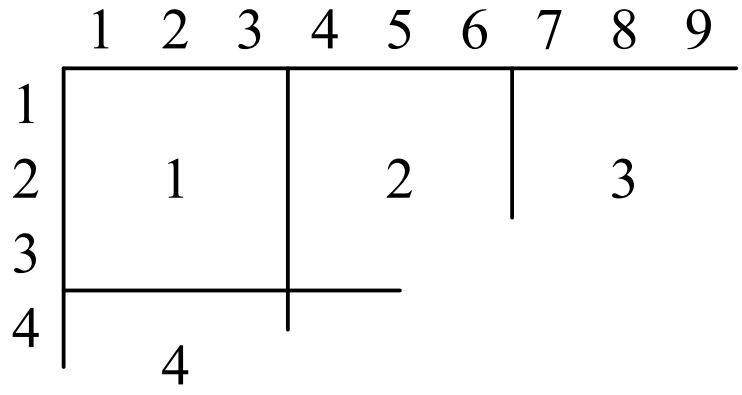


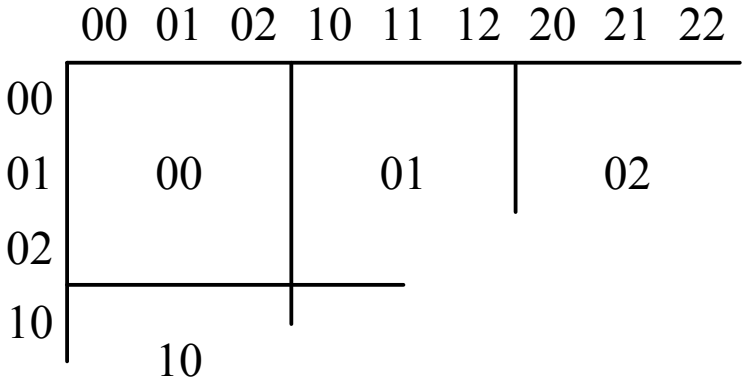
4. Hardwareimplementierung

Adresslogik

- **Vermeiden von Mul und Div**
  - nötig zur Berechnung des Blocks aus Zeile und Spalte
  - Rechnen im Zahlensystem mit Basis B=N

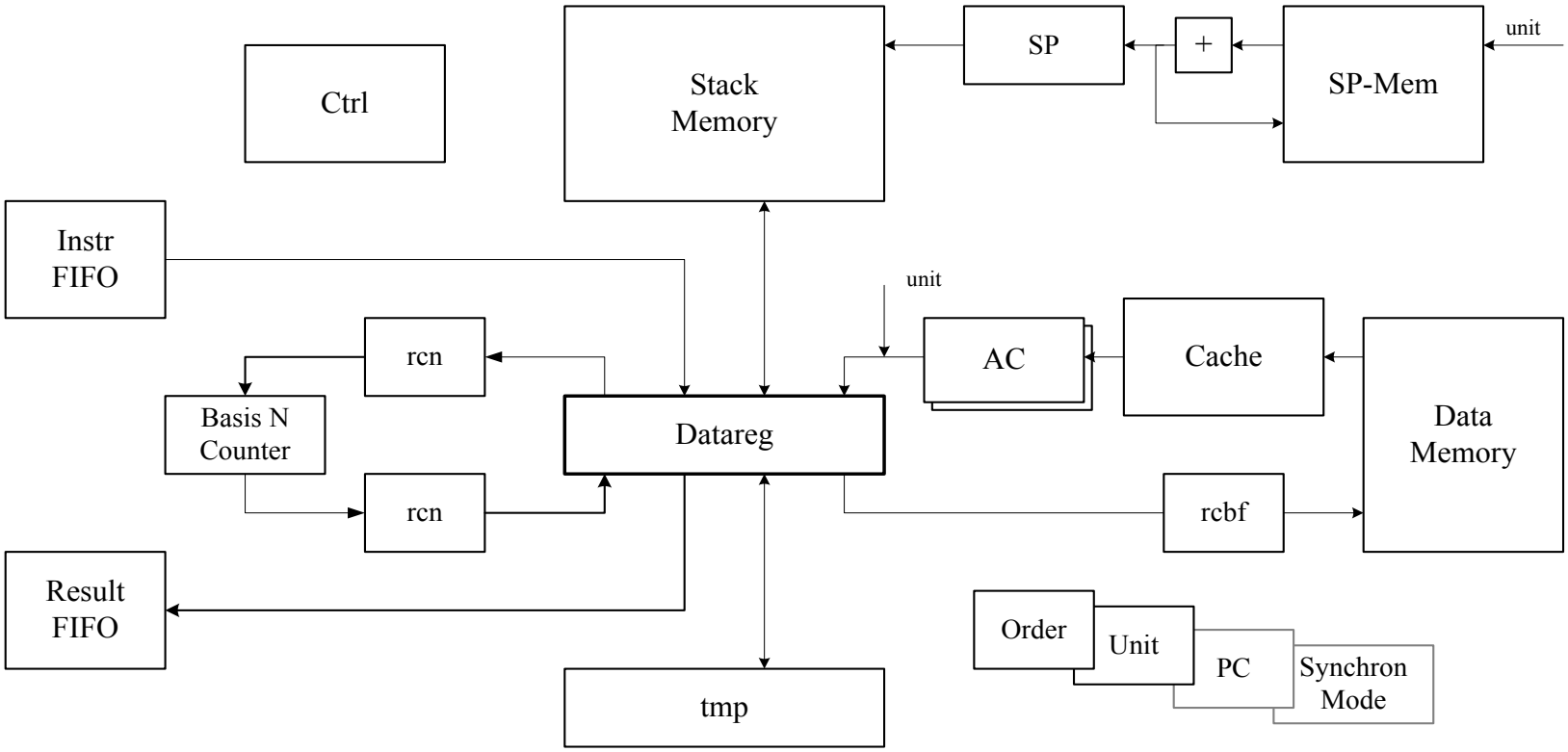






4. Hardwareimplementierung

Entwurf



5. Messungen

Prolog vs. Backtracking

- **N = 3**
- **GNU Prolog auf Core2Duo E7400 @ 2,8 GHz**
- **Backtracking auf Spartan3E SK @ 50 MHz**

Prolog	0,688 s
Backtracking	0,00079 s

6. To Do

bis Februar/März

- **Diskussion Parallelisierung / Partitionierung**
- **weitere Löser**
  - nur CSP
  - nichtdeterministisch
- **Benchmarks**
- **Messungen**

Literatur

- Kumar, Vipin: Algorithms for Constraint Satisfaction Problems: A Survey. In: AI Magazine 13 (1992), S. 32-44
- Mackworth, Alan K.: Consistency in Networks of Relations. In: Artificial Intelligence 8(1) (1977): S. 99-118
- Crawford, Aranda, Castro, Monfroy: Using Constraint Programming to solve Sudoku Puzzles. In: Third 2008 International Conference on Convergence and Hybrid Information Technology (2008), S. 926-931