TECHNISCHE UNIVERSITÄT DRESDEN

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

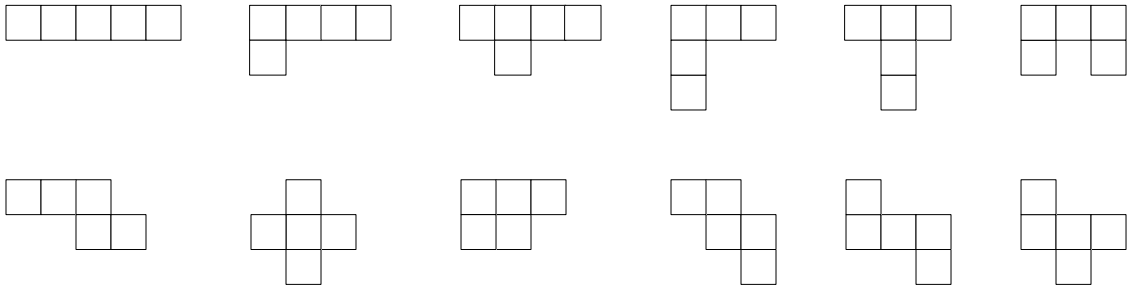
Vortrag zum Diplom

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

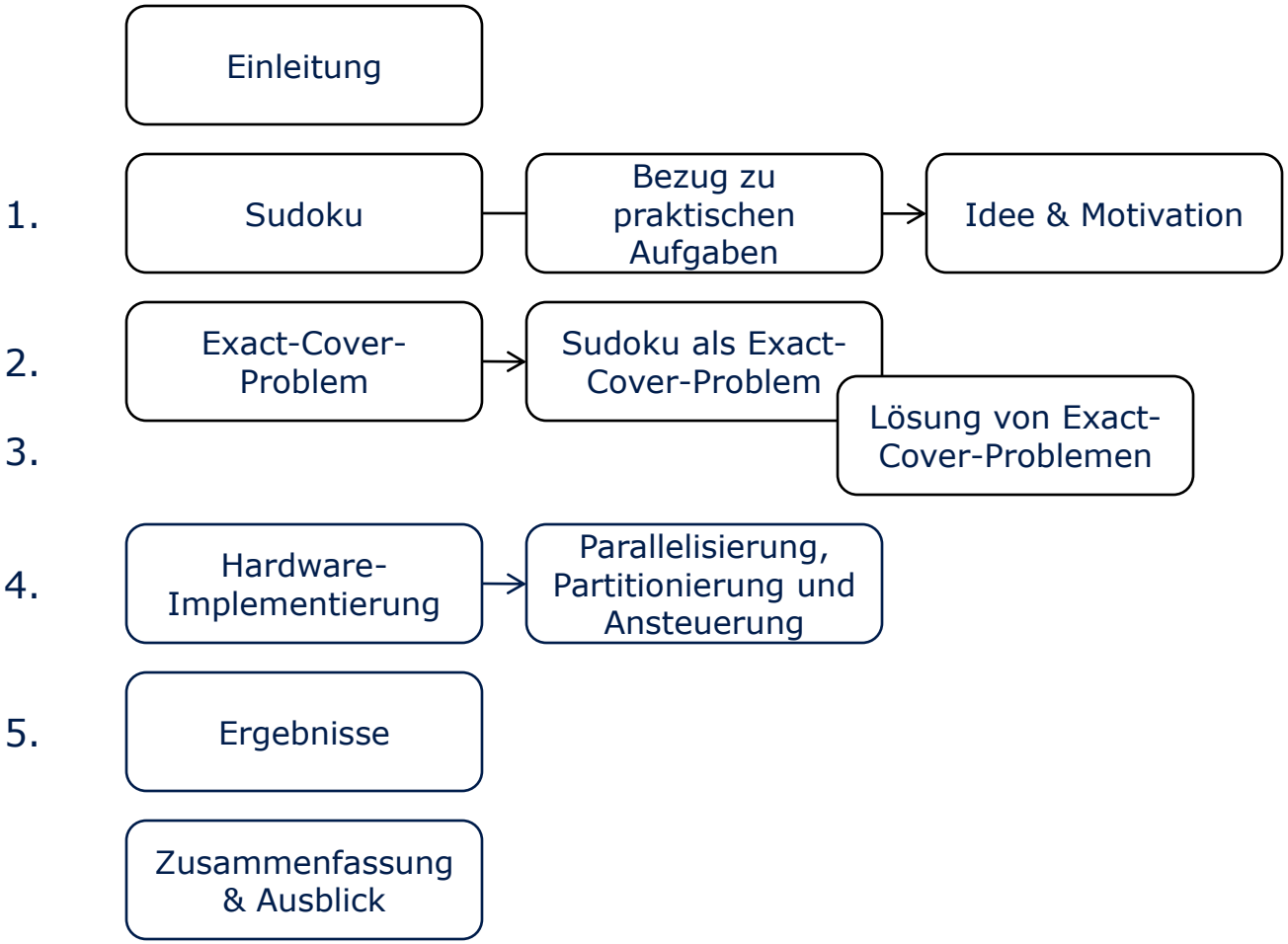
Michael Dittrich, michael-dittrich@mailbox.tu-dresden.de

Dresden, 14.04.2010

Einleitung: Pentomino



Gliederung


Einleitung
1.
Sudoku
Bezug zu praktischen Aufgaben
Idee & Motivation
2.
Exact-Cover-Problem
Sudoku als Exact-Cover-Problem
3.
Lösung von Exact-Cover-Problemen
4.
Hardware-Implementierung
Parallelisierung, Partitionierung und Ansteuerung
5.
Ergebnisse
Zusammenfassung & Ausblick

1. Motivation

Sudoku ist ein Zahlenpuzzle

			4
	3		
		2	
1			

2. Ordnung

			4					1
	4		2			5		
	7	9					3	
6			5	8		1		7
	8		6					
	5				7	3		
2	6						4	
		8			4			

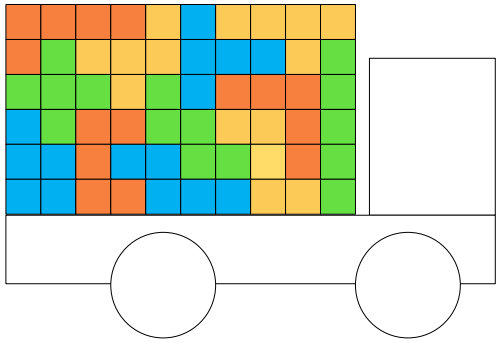
3. Ordnung
FPT'09 (03b)

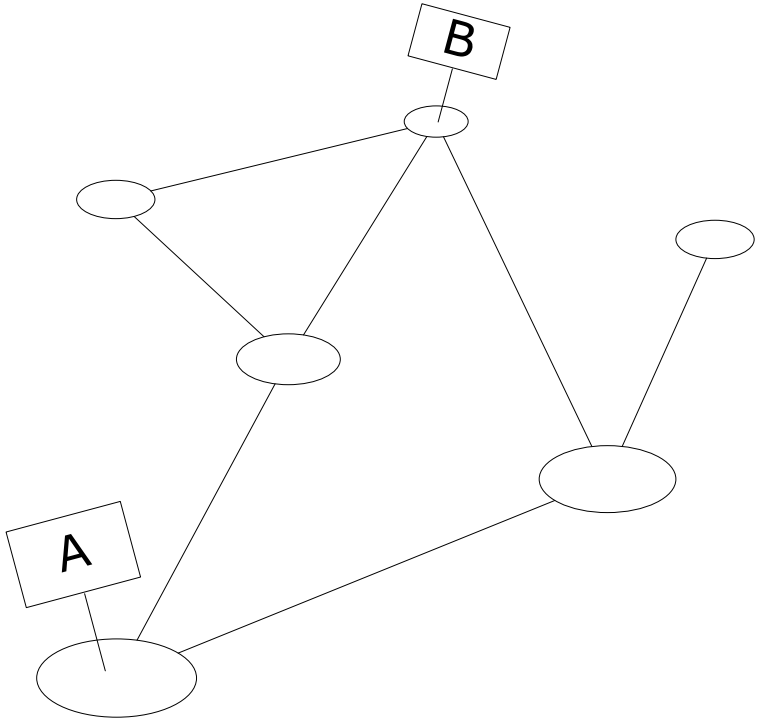
2	4		6		7		13				8			16	12
			8			12		9			16				6
			11		4				7	12			3	13	
		14	15					2			11		8		
5		1				10								6	
4		8	3					13	2	16	1		11	14	
		9			13		1		8						
					9	2	5			14				4	7
						8				6	12			3	4
	3	13		14			10								
								1		13	10	8		2	
		16										9	5		14
11				16			7				3	14			
	14						12		5			15		11	1
12					5			7	9	1	2		16		
			13				2	8				4	7		5

4. Ordnung
FPT'09 (04b)

1. Motivation

praktische Probleme

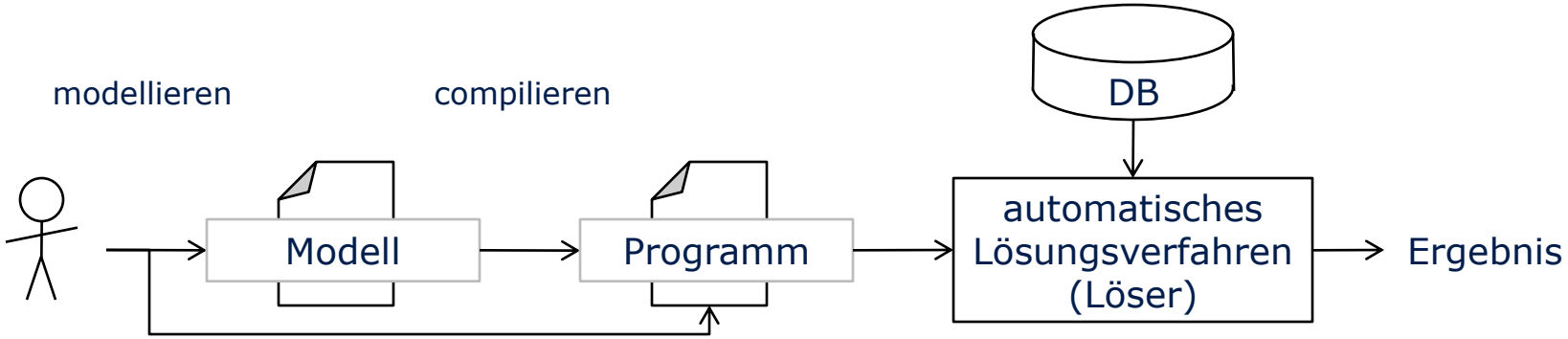




1. Motivation

Idee

- Modellierung in einfacher Beschreibungssprache
- Ziel ist Lösung des Problems ohne Angabe eines Lösungsalgorithmus!



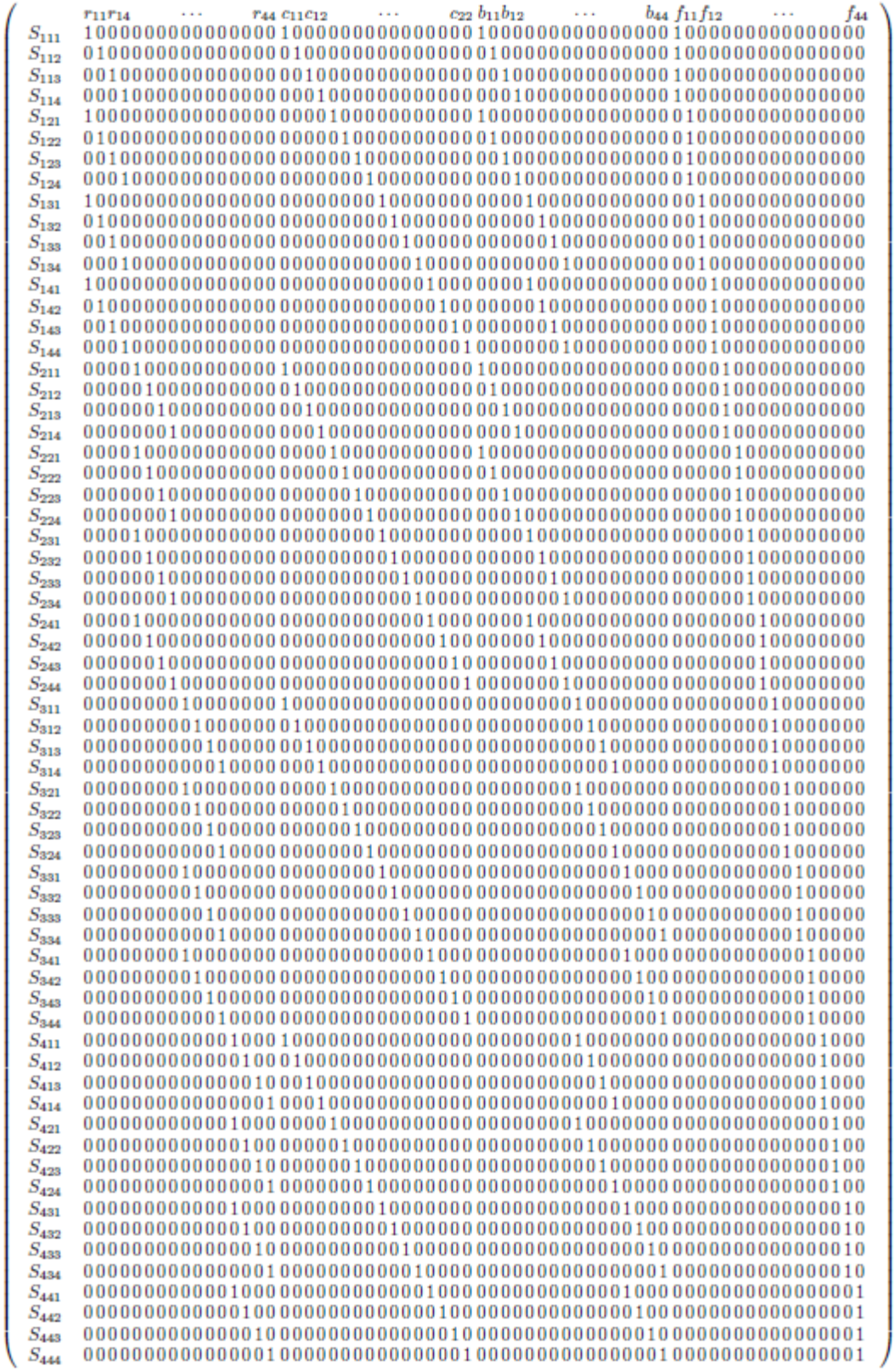
2. Exact Cover Problem

Beispiel

- S = {{A, B}, {B, C}, {C}}

	A	B	C
S1	1	1	
S2		1	1
S3			1

- Mengen entsprechen „Lösungsschritten“
- Variablen markieren sich ausschließende Lösungsschritte („Bedingungen“)


$r_{11}r_{14}$ ⋯ r_{44} $c_{11}c_{12}$ ⋯ c_{22} $b_{11}b_{12}$ ⋯ b_{44} $f_{11}f_{12}$ ⋯ f_{44}
S_{111} S_{112} S_{113} S_{114} S_{121} S_{122} S_{123} S_{124} S_{131} S_{132} S_{133} S_{134} S_{141} S_{142} S_{143} S_{144} S_{211} S_{212} S_{213} S_{214} S_{221} S_{222} S_{223} S_{224} S_{231} S_{232} S_{233} S_{234} S_{241} S_{242} S_{243} S_{244} S_{311} S_{312} S_{313} S_{314} S_{321} S_{322} S_{323} S_{324} S_{331} S_{332} S_{333} S_{334} S_{341} S_{342} S_{343} S_{344} S_{411} S_{412} S_{413} S_{414} S_{421} S_{422} S_{423} S_{424} S_{431} S_{432} S_{433} S_{434} S_{441} S_{442} S_{443} S_{444}

3. Lösen von Exact Cover Problemen

Beispiel: Algorithmus

	1	2	3	4
a				
b		3		
c			2	
d	1			

↓

	1	2	3	4
a		1		
b		3	1	
c	3	4	2	1
d	1	2		

≅

a1:2	1000000	1000000	1000000	1000000
a1:4	0010000	0100000	0100000	1000000
a3:3	0100000	0010000	0001000	0100000
a3:4	0010000	0001000	0000100	0100000
a4:2	1000000	0000100	0100000	0010000
a4:3	0100000	0000010	0001000	0010000
a4:4	0010000	0000001	0000100	0010000
b1:2	0001000	1000000	1000000	0001000
b1:4	0000100	0100000	0100000	0001000
b4:2	0001000	0000100	0010000	0000100
b4:4	0000100	0000001	0000100	0000100
d3:3	0000010	0010000	0000010	0000010
d3:4	0000001	0001000	0000001	0000010
d4:3	0000010	0000010	0000010	0000001
d4:4	0000001	0000001	0000001	0000001

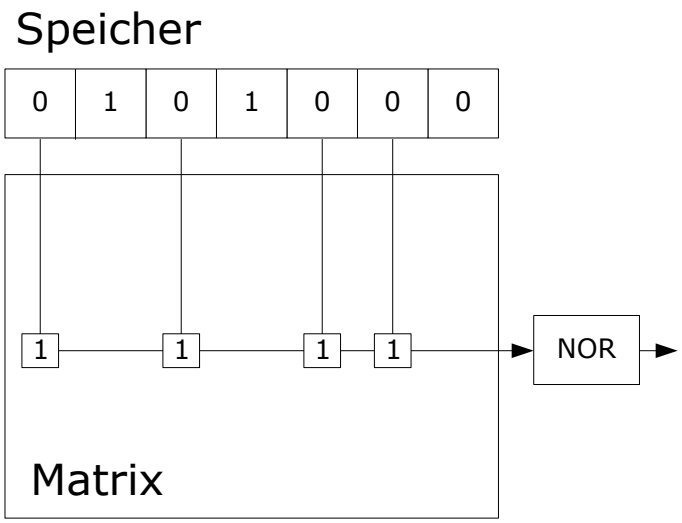
3. Lösen von Exact Cover Problemen

Beispiel: Suchbaum

			4					1
	4		2			5		
	7	9					3	
6			5	8		1		7
	8		6					
	5				7	3		
2	6						4	
		8			4			

4. Hardwareimplementierung

Datenstruktur

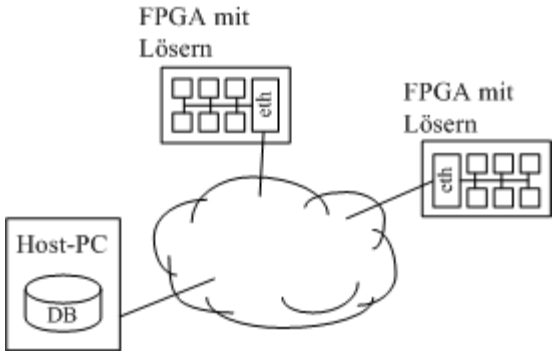


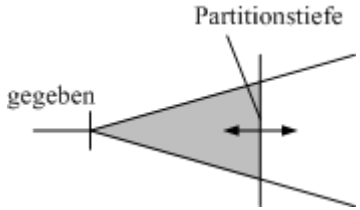
Vorteile:

1. minimaler Speicherbedarf für Spielzustand
2. automatischer Ausschluss der Kandidaten
3. alte Spielzustände sind rekonstruierbar

4. Hardwareimplementierung

Parallelisierung, Steuerung, Partitionierung der Sudokus





5. Ergebnisse

Synthese

Konfiguration	MHz	Register	LUTs
SISD	100	3319	3651
SISD_2048	100	3333	3766
MIMD8_256	100	10505	18931

5. Ergebnisse

Suche nach allen Lösungen, Puzzle 05b, ML505

Konfiguration	Laufzeit (Summe)	Laufzeit (theoretisch)	Speedup
no resume	745,1 s		1/ 2,23
no skip	1011,8 s		1/ 3,34
no resume & no skip	2856,4 s		1/ 8,56
SISD	**333,5 s**		
SISD_2048	178,6 s		1,87
SISD_2048_2048	178,1 s		1,87
MIMD8_256	205,3 s	25,7	6,96
MIMD8_256_256	204,7 s	25,6	6,98

5. Ergebnisse

Vergleich mit Prolog, Zeiten in ms

- Prolog Model, gelöst mit GNU-Prolog,
  Intel(R) Core(TM) 2 Duo CPU, E7400, 2.79GHz
- SISD Löser, 32 Wörter Datencache, 256 Wörter Stackcache,
  Xilinx(R) ML505(TM), 100MHz

Konfiguration	03a	04a	05a	06a	07a	03b	04b	05b
E7400	198	318	18s	250	448	187	11,6s	1:35h
SISD_32_256	0,25	1,1	8,5	2,8	7,1	7,1	1,1	231
Speedup	792	279	2154	25	63	26	10133	25

	TU Delft	University Madrid	TU Crete	SISD_1024
03a	0,020821 s	0,008600 s		0,000257 s
04a	0,221379 s	0,024761 s		0,001306 s
05a		0,060281 s		0,009064 s
06a	0,115347 s	0,117894 s	0,145 s	0,002838 s
07a	0,211343 s	0,224523 s	0,223 s	0,007863 s
08a	0,096424 s	0,058375 s	0,658 s	0,016489 s
09a		0,300835 s	5,662 s	0,075043 s
10a		0,322603 s	19,780 s	0,116940 s
11a		0,381182 s	147,490 s	0,183400 s
12a		-	557,520 s	0,558018 s
13a		-		0,802226 s
14a		-		2,017806 s
03b	0,012460 s	0,009762 s		0,003178 s

Zusammenfassung

- **Löser ist schneller als Prolog und andere HW-Löser**
- **Aber:**
  - a-Puzzles sind sehr einfach. schwere b-Puzzles sind nicht lösbar (<12h)
  - leeres Sudoku N=7 besetzen nicht möglich (<12h)
- **Exponentialkurve offenbar sehr stark gekrümmt**
  - höhere Lösbarkeitsklasse nicht durch Beschleunigung nicht erreichbar

1. **Durch Auswahlheuristik die Suche begrenzen.**
2. **Durch mehr „Intelligenz" den Suchbaum verkleinern.**

Literatur

- [BTAS09] van der Bok, Keed; Taouil, Mottaqiallah; Afratis, Panagiotis; Sourdis, Ioannis: The TU Delft Sudoku Solver on FPGA. In: 2009 International Conference on Field-Programmable Technology, 2009
- [GOR09] Gonzalez, Carlos; Olivito, Javier; Resano, Javier: An Initial Specic Processor for Sudoku Solving. In: 2009 International Conference on Field-Programmable Technology, 2009
- [Knu00] Knuth, Donald E.: Dancing Links. In: Millenial Perspectives in Computer Science (2000)
- [Kum92] Kumar, Vipin: Algorithms for Constraint Satisfaction Problems: A Survey. In: AI Magazine 13 (1992)
- [Mai08] Mailer, Glen: A Guess-Free Sudoku Solver. Master Thesis, 2008
- [MSSD09] Malakonakis, Pavlos; Smerdis, Miltiadis; Sotiriades, Euripides; Dollas, Apostolos: An FPGA-Based Sudoku Solver based on Simulated Annealing Methods. In: 2009 International Conference on Field-Programmable Technology, 2009