Analyse von Ansätzen zur Beschleunigung von SAT - Lösern durch dedizierte Hardware Komponenten

E. Zenker

9. November 2011

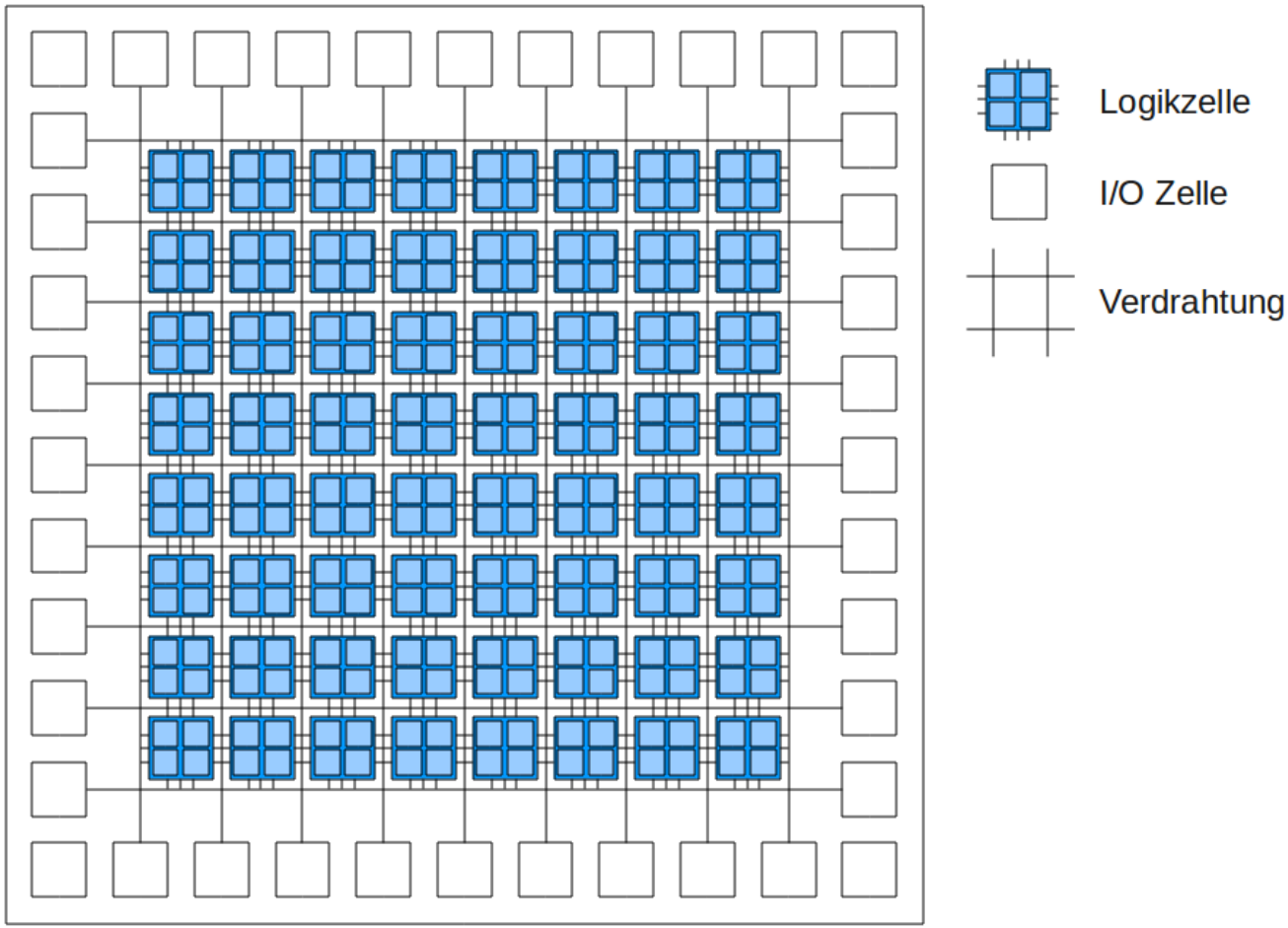
Gliederung

1. Field Programmable Gate Array - FPGA
2. Satisfiability Testing - SAT
3. FPGA-SAT Entwurf
4. Resultate
5. Fazit
6. Ausblick

FPGA - Field Programable Gate Array

- Logikzellen als Matrix angeordnet
  - Mehre Lookup Tables (LUTs) pro Logikzelle
  - Pro LUT ein FlipFlop
- I/O Zellen als spezielle Logikzellen
- Programmierbare Verbindungen zwischen Logikzellen
- Spezielle Hard Macros
  - Multiplizierer, Ethernet, Speicher Kontroller etc.
- zusätzlich Block-RAM auf FPGA integriert

FPGA - Field Programable Gate Array



Aussagenlogik

Syntax

- Variablen der Aussagenlogik

  $V = \{ 1, 2, ..., n \}$

- Operationen

  Negation $\neg$, Konjunktion $\wedge$, Disjunktion $\vee$

- Literale sind Variablen oder negierte Variablen

- Klausel

  Distjunktion von Literalen $C = [l_1, l_2, l_3, ...]$

- Unit Klausel

  Klausel, welche nur aus einem Literal besteht $C = [l_1]$

Aussagenlogik

Syntax

- Formel

  Konjunktion von Klauseln $F = \langle C_1, C_2, ... \rangle$

- Gruppe

  Menge von Klauseln $G = \{C_1, C_2, ...\}$, wobei jede Variable maximal einmal vorkommt

- Gruppierung einer Formel

  $group : F \to \mathcal{G}$, weist jeder Klausel einer Formel eindeutig eine Gruppe zu.

Aussagenlogik

Semantik

- Interpretation $\mathcal{I}$

  $\mathcal{I} : var(F) \rightarrow \mathcal{W}$ mit $\mathcal{W} = \{\top, \bot\}$

- Partielle Interpretation J
  - Bildet nicht alle Variablen einer Formel auf einen Wahrheitswert ab
  - Darstellung als Liste von Literalen: $J = (l_1, l_2, ..., l_n)$
- Ein Literal ist erfüllt, wenn es in J enthalten ist
- Eine Klausel C ist erfüllt, wenn mindestens ein Literal in C erfüllt ist.
- Eine Formel F ist erfüllt, wenn alle Klauseln in F erfüllt sind

Aussagenlogik

Semantik

- Redukt $F|_J$
  - Alle Klauseln, welche mindestens ein erfülltes Literal aus J enthalten, werden aus F entfernt.
  - Alle Literale, welche negiert in J vorkommen, werden aus ihren Klauseln in F entfernt.

Lösen des SAT-Problems

- Gegeben:
  - Eine Formel F

    $\mathrm{F} = \langle\, [1,\, 3],\, [\neg 2,\, \neg 5,\, \neg 6],\, [\neg 1,\, \neg 4,\, 6],\, [\neg 1, \neg 2, \neg 4,\, 5],\, [\neg 1,\, 2] \rangle$
  - Eine leere partielle Interpretation J
- Gesucht: Ein Modell für die Formel, wenn vorhanden

  $J = (1, 2, 3, \neg 4, 5, \neg 6)$
- Frage: Wie kann ein solches Modell gefunden werden ?

Lösen des SAT-Problems

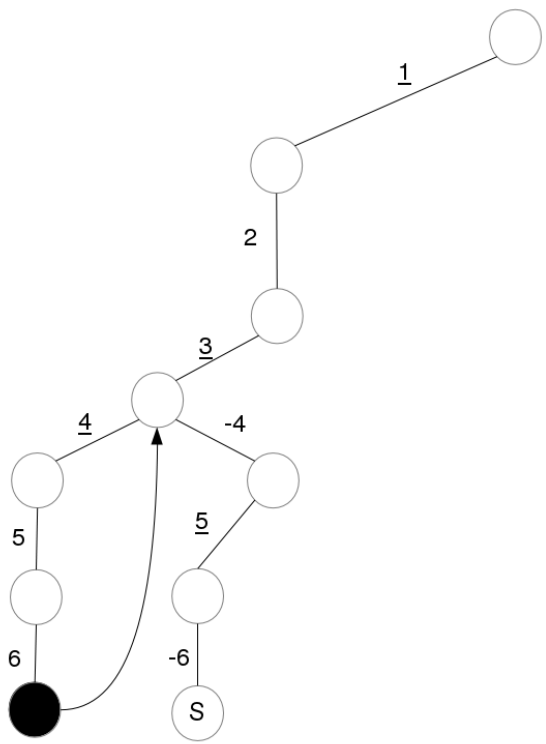
Davis Putman Logeman Loveland (DPLL)

1. Entscheidung für eine freie Variable
   - keine freien Variablen $\rightarrow$ SAT
2. Bildung des Redukts $F|_J$ (Inferenz)
   - Leere Klausel (Konflikt):
     - keine Entscheidungsliteral in $J \rightarrow$ UNSAT
     - Entfernen der Literale aus J bis zur letzten Entscheidung und Negieren der letzten Entscheidung (Rücksprung)
     - Algorithmus bei 2. fortführen
   - Unit Literale werden J hinzugefügt und es geht bei 1. weiter

Lösen des SAT-Problems

Davis Putman Logeman Loveland (DPLL)

- $F = \langle [1, 3], [\neg 2, \neg 5, \neg 6], [\neg 1, \neg 4, 6], [\neg 1, \neg 2, \neg 4, 5], [\neg 1, 2] \rangle$

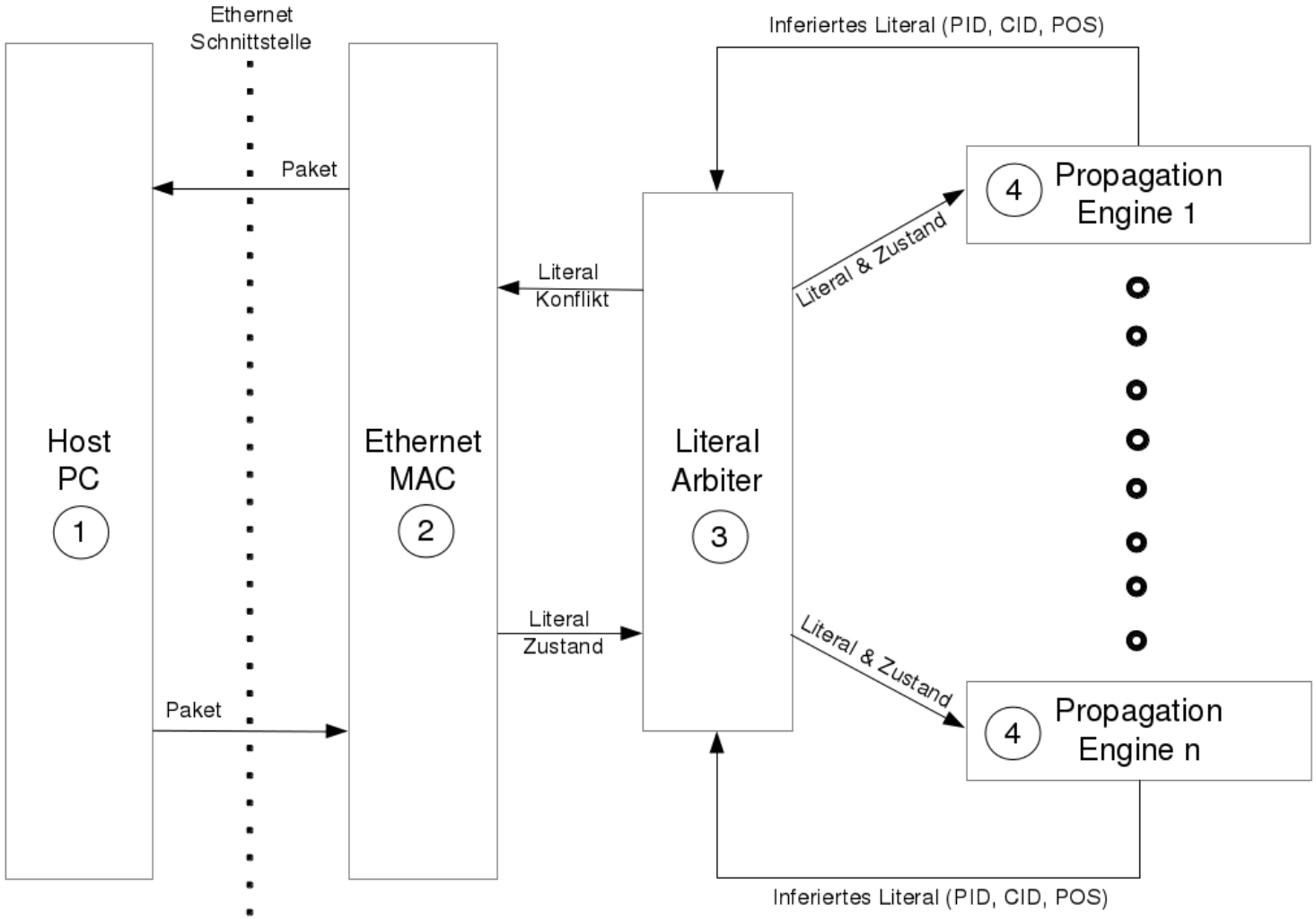


FPGA-SAT

- Hybrider SAT-Löser
  - Arbeitsteilung zwischen Host-PC und FPGA
  - Kommunikation über Ethernet
  - Grundlage ist DPLL Algorithmus
- Host-PC
  - Entscheidungsheuristic
  - Rücksprungberechnung
- FPGA
  - Inferenz von Literalen
  - Konflikterkennung

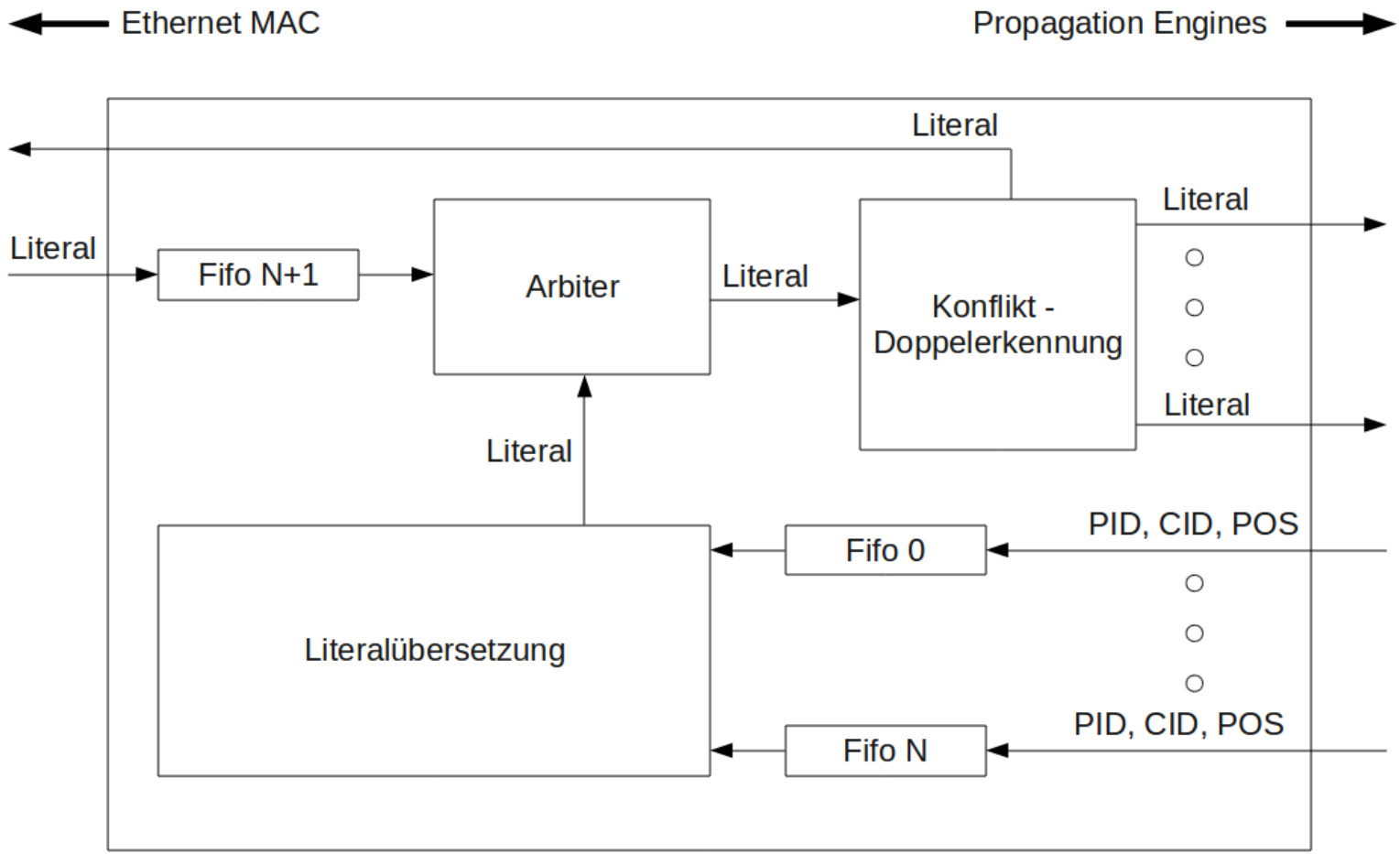
FPGA-SAT

Systemüberblick



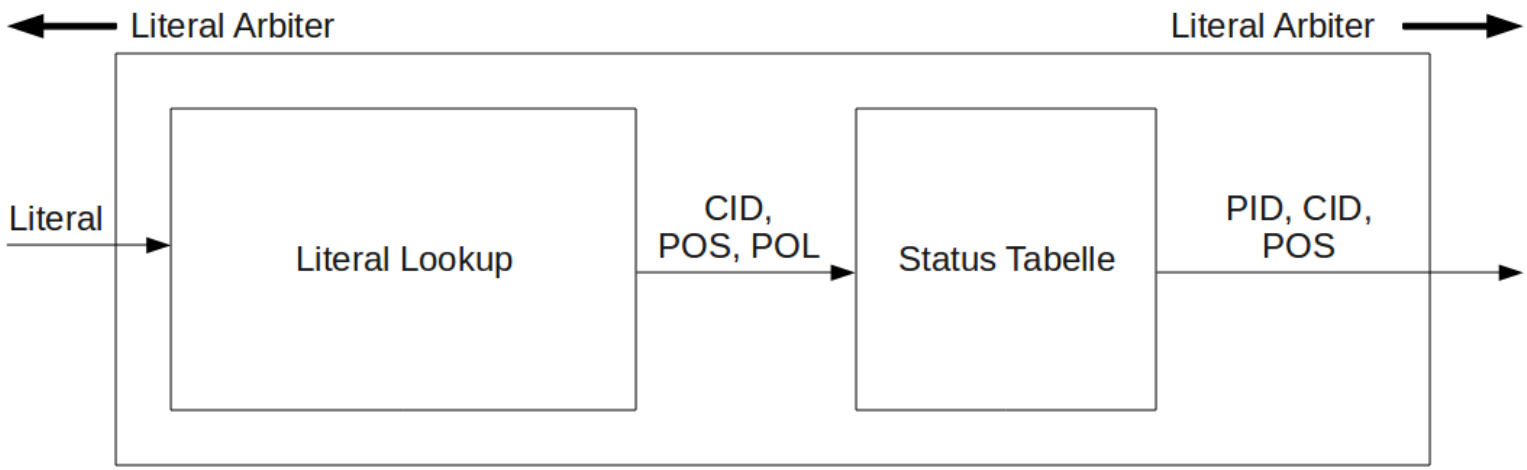
FPGA-SAT

Literal - Arbiter



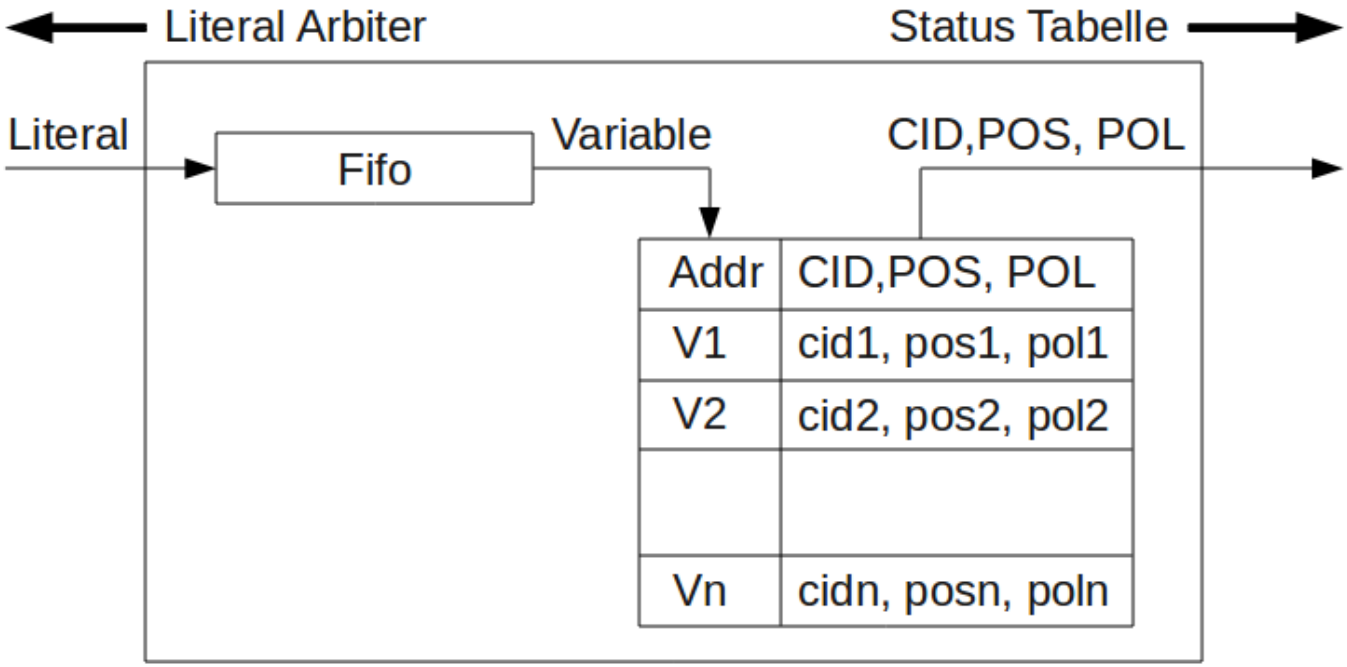
FPGA-SAT

Propagation - Engine



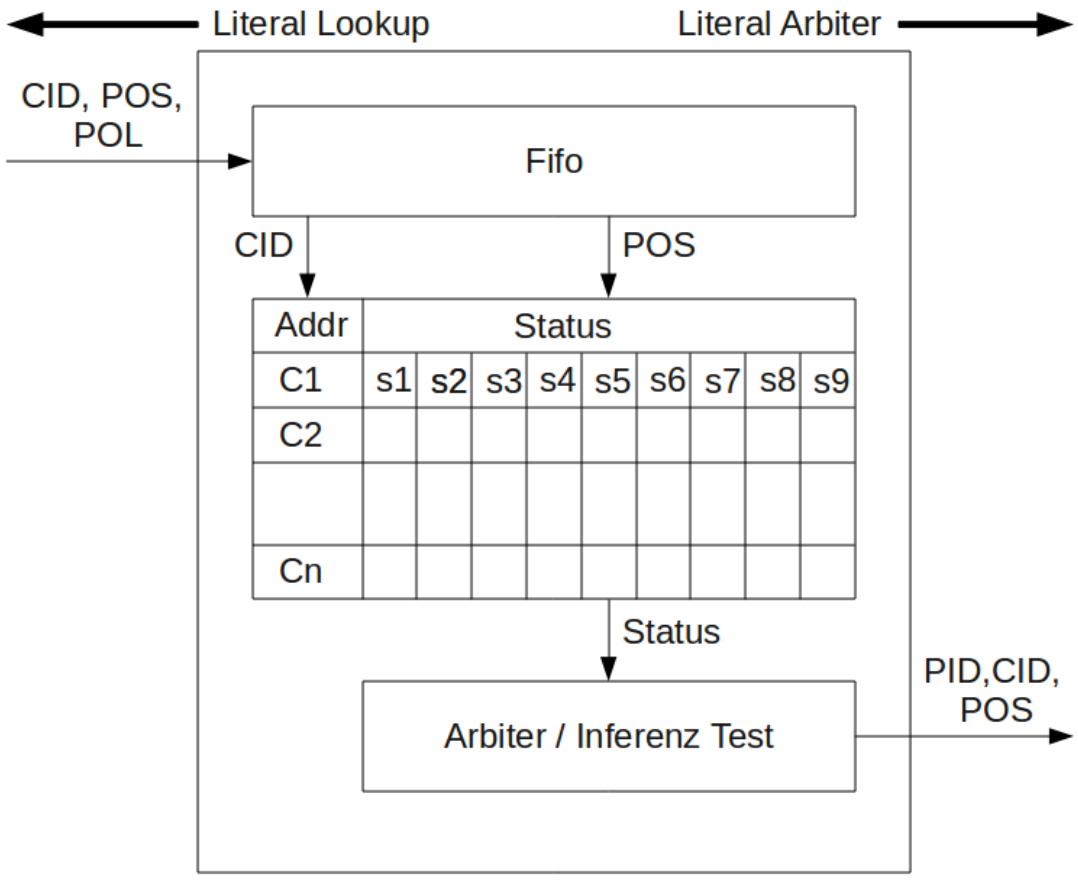
FPGA-SAT

Literal - Lookup



FPGA-SAT

Status - Tabelle



Resultate

Synthese

- Synthese für Xilinx Virtex 5 XC5VLX50T (ML505 Entwicklerboard)
- Formeln in 9-SAT
- 256 Variablen
- 32 Propagation Engines mit je 128 Klauseln (4096 Klauseln)
- Getaktet mit 125 Mhz

Resultate

Experimente

- Vergleich von Software mit Hybrid Lösung
- Folgende Problem-Instanzen wurden ausgewählt:
  - **Testfall 1 :** Parallele Inferenz
  - **Testfall 2 :** Serielle Inferenz
  - **Testfall 3 :** Bereits vorhandenen Instanzen

Resultate

Parallele Inferenz

- $F_{Test1} =$

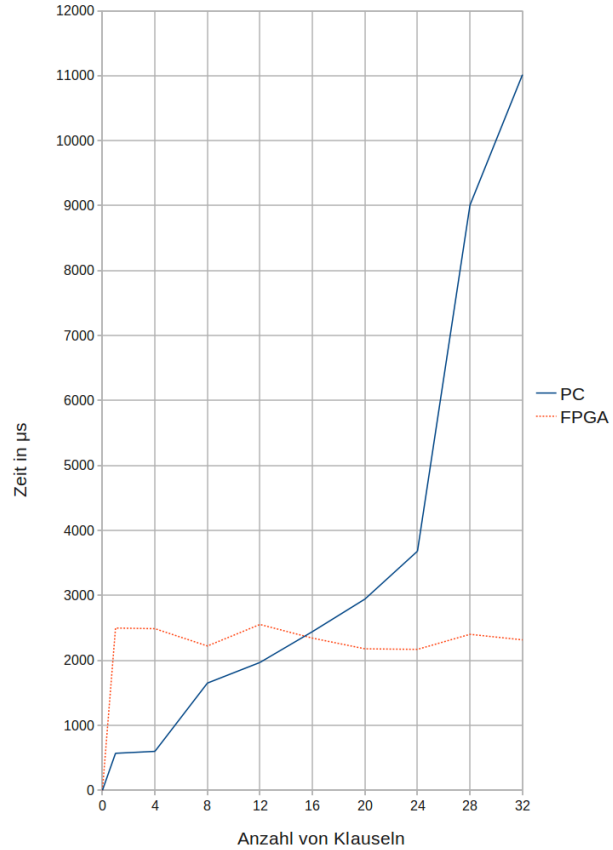
  $\langle [1,2], [1,3], ..., [1,32], [1,33] \rangle$

- Durch erfüllen von

  $I = \neg 1$ werden

  mehrere Unit Klauseln

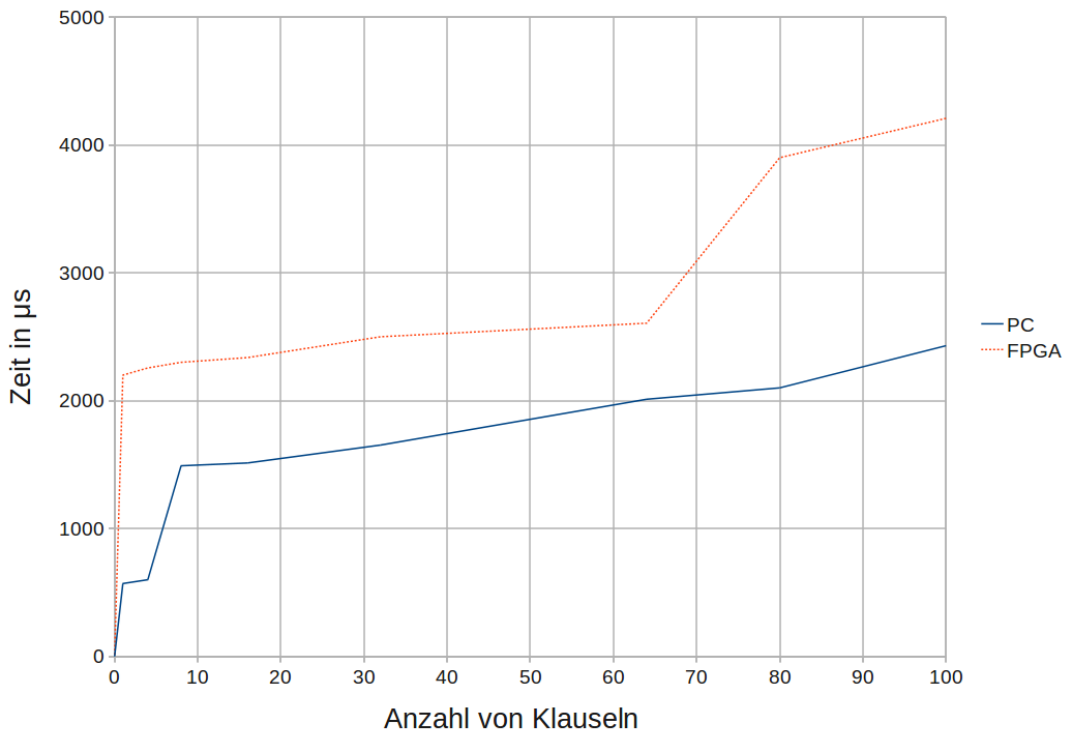
  erzeugt



Resultate

Serielle Inferenz

- $F_{Test2} = \langle [1, 2], [\neg 2, 3], [\neg 3, 4], [\neg 4, 5][\neg 5, 6], ..., [\neg 100, 101] \rangle$
- Durch erfüllen von $l = \neg 1$ wird eine Inferenzkette erzeugt



Resultate

Vorhandene Instanzen

- Eckdaten ausgewählter Problem-Instanzen

Instanz	Variablen	Klauseln	Gruppen
anomaly.cnf	48	261	22
flat-easy-1.cnf	90	300	8
289-sat-4x8.cnf	128	896	25

Resultate

Vorhandene Instanzen

- Ausgewählter Problem-Instanzen im Vergleich

Instanz	Laufzeit PC	Laufzeit FPGA	Pakete	Sendezeit
anomaly.cnf	6,1 ms	10,2 ms	17	5,1 ms
flat-easy-1.cnf	8,2 ms	22,0 ms	53	15,9 ms
289-sat-4x8.cnf	22,6 ms	60,7 ms	129	38,7 ms

Fazit

- Pro FPGA-SAT
  - Bessere parallele Verarbeitung von Literalen
  - Gleiche Leistung bei serieller Verarbeitung von Literalen
- Kontra FPGA-SAT
  - Große Kommunikationslatenzen durch Ethernet-Schnittstelle
  - Lösbare Problem-Instanzen sind sehr klein

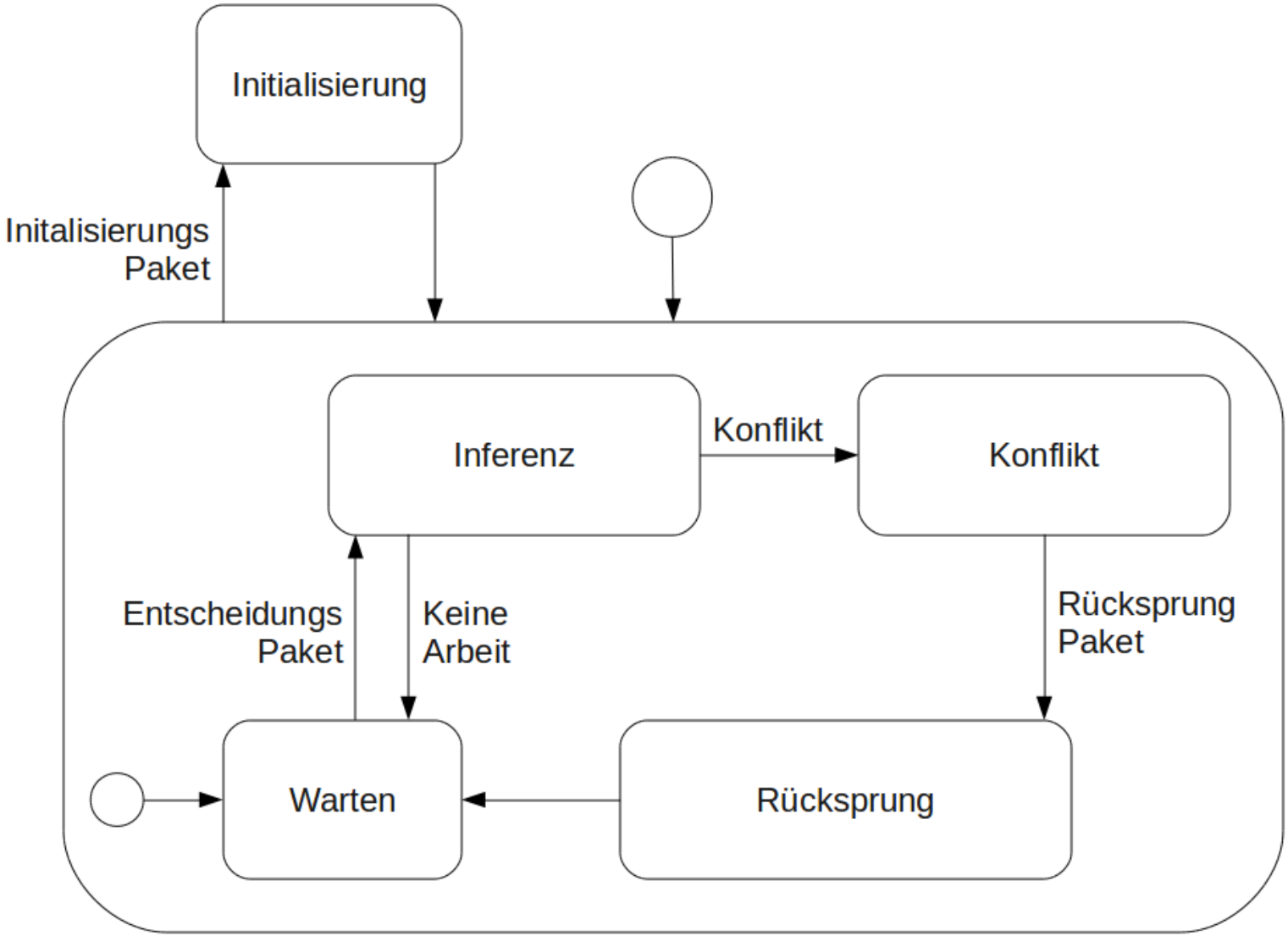
Ausblick

- Verbesserung der Kommunikationslatenz durch schnellere Schnittstelle (HT, PCIe)
- FPGA-SAT-Löser ohne Host-PC
- Größere Problem-Instanzen durch größere Speicher (SRAM auf ML505) und verbesserten Literal Lookup
- Nutzung moderner Lösungtechniken (CDCL)

Vielen dank für Ihre Aufmerksamkeit

Synthese Ergebnisse

Logikelement	Benutzt	Verfügbar	Ausnutzung
LUTs	10053	28800	34%
davon LUTs als Logik	8348	-	-
davon LUTs als Speicher	1698	-	-
FlipFlops	5701	28800	19%
Block-RAM	53	60	88%
davon 18 Kbit Block-RAM	70	-	-
davon 36 Kbit Block-RAM	18	-	-


Initialisierung
Initalisierungs
Paket
Inferenz
Konflikt
Konflikt
Entscheidungs
Paket
Keine
Arbeit
Rücksprung
Paket
Warten
Rücksprung