Analyse der Nutzbarkeit der Carry-Chain-Strukturen moderner FPGAs für kombinatorische Funktionen

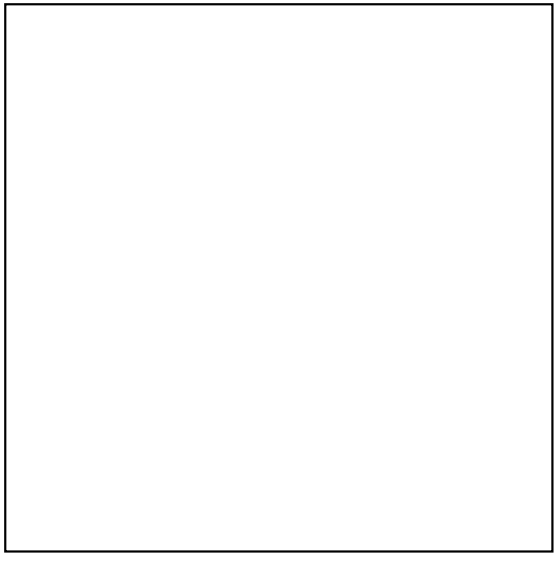
Julia Daniel

Institut für Wissenschaftliches Rechnen
Institut für Technische Informatik

14. April 2011

FPGA

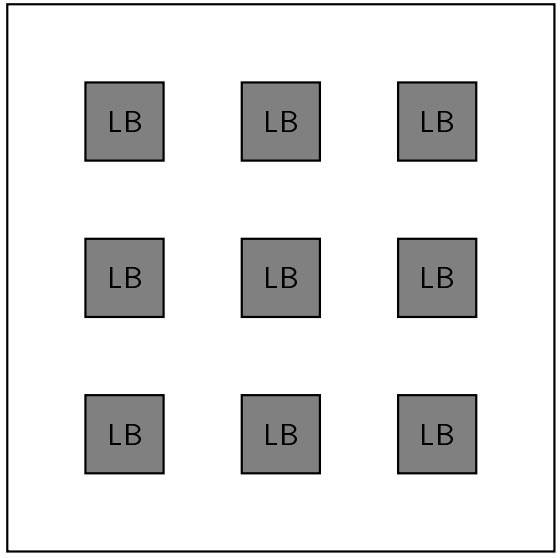
Ein field-programmable Gate-Array (FPGA) ist ein integrierter Schaltkreis. Hauptkomponenten:



FPGA

Ein field-programmable Gate-Array (FPGA) ist ein integrierter Schaltkreis. Hauptkomponenten:

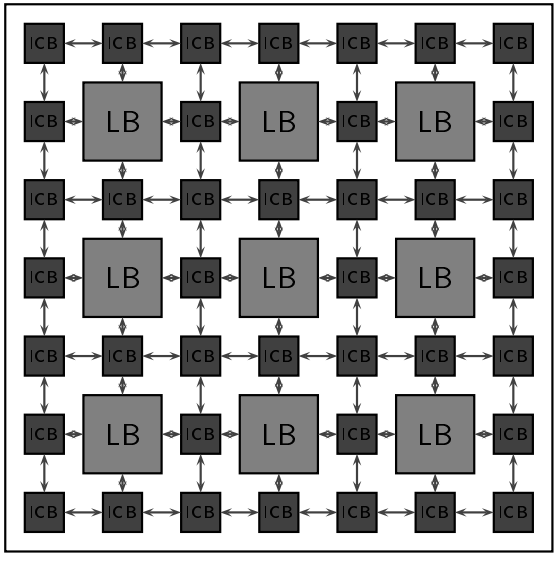
- Matrix konfigurierbarer Logikblöcke



FPGA

Ein field-programmable Gate-Array (FPGA) ist ein integrierter Schaltkreis. Hauptkomponenten:

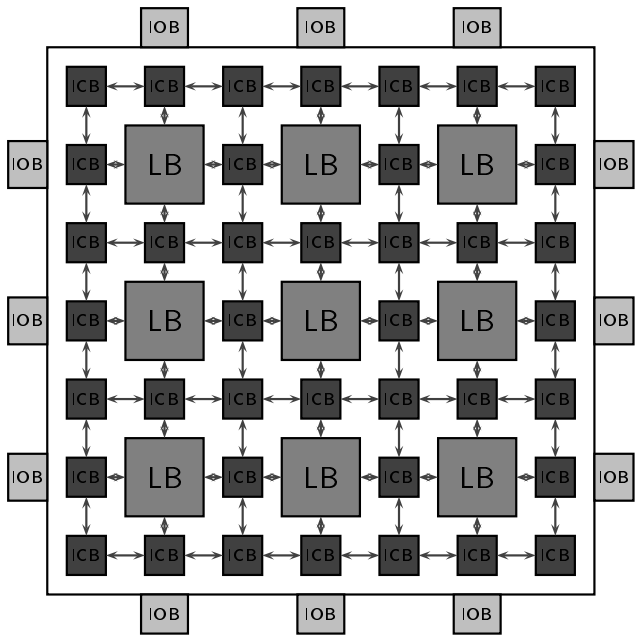
- Matrix konfigurierbarer Logikblöcke
- Routing-Matrix



FPGA

Ein field-programmable Gate-Array (FPGA) ist ein integrierter Schaltkreis. Hauptkomponenten:

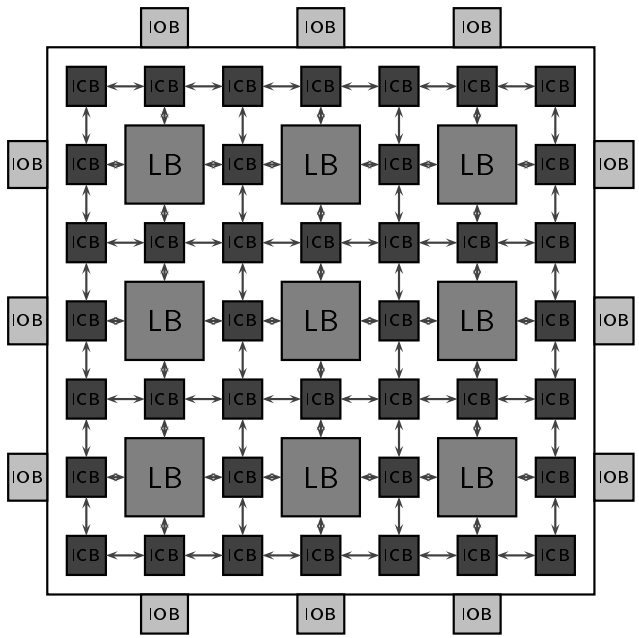
- Matrix konfigurierbarer Logikblöcke
- Routing-Matrix
- Input-/Outputblöcke



FPGA

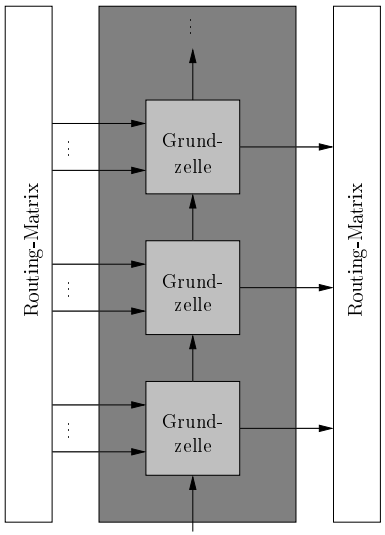
Ein field-programmable Gate-Array (FPGA) ist ein integrierter Schaltkreis. Hauptkomponenten:

- Matrix konfigurierbarer Logikblöcke
- Routing-Matrix
- Input-/Outputblöcke
- Anwendungsspezifische Komponenten z.B. Addierer, Mikroprozessoren



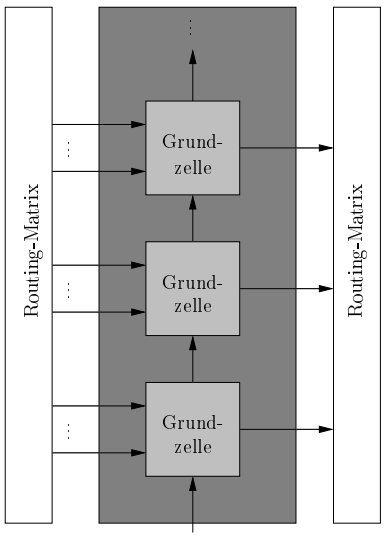
Logikblöcke

- sind in Grundzellen arrangiert

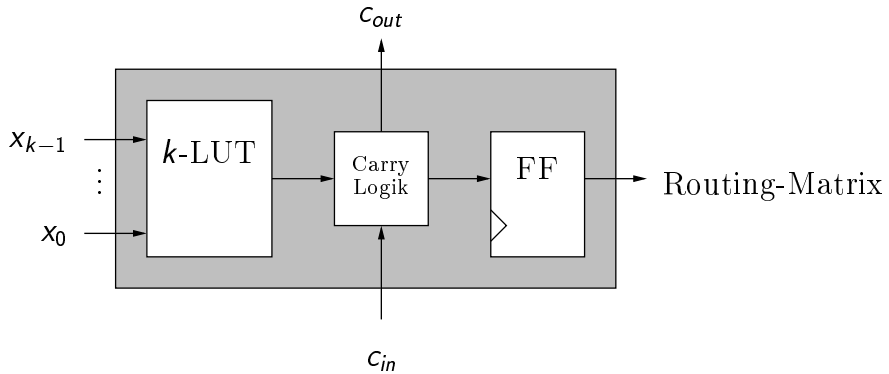


Logikblöcke

- sind in Grundzellen arrangiert



- jede Grundzelle beinhaltet eine Look-Up-Table (LUT), Carry-Logik und ein Speicherelement (FF)



Gegeben: Summanden $a = (a_1, a_0), b = (b_1, b_0)$
eingehender Übertrag $c_0 = 0$

Gesucht: Summe $s = (s_1, s_0)$, ausgehender Übertrag c_2

Gegeben: Summanden $a = (a_1, a_0), b = (b_1, b_0)$
eingehender Übertrag $c_0 = 0$

Gesucht: Summe $s = (s_1, s_0)$, ausgehender Übertrag c_2

funktionelle Beschreibung (HDL)

Gatterdarstellung

$$c_{i+1} = a_i b_i + c_i(a_i \oplus b_i)$$
$$s_{i+1} = a_i \oplus b_i \oplus c_i$$

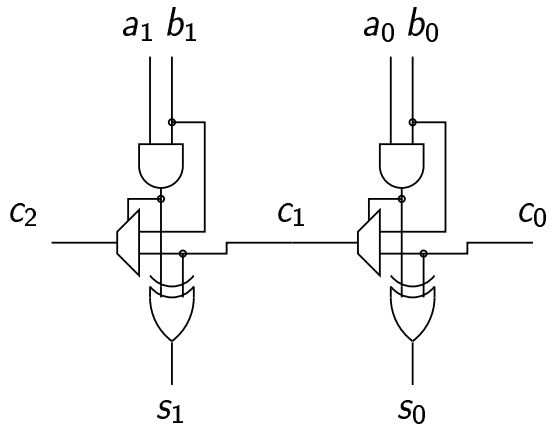
Gegeben: Summanden $a = (a_1, a_0), b = (b_1, b_0)$
eingehender Übertrag $c_0 = 0$

Gesucht: Summe $s = (s_1, s_0)$, ausgehender Übertrag c_2

funktionelle Beschreibung (HDL)

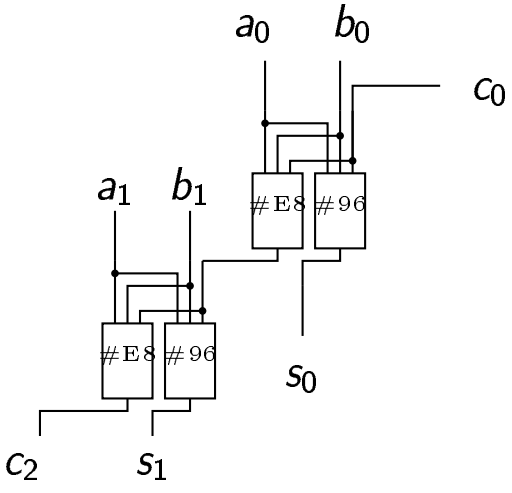
$$c_{i+1} = a_i b_i + c_i(a_i \oplus b_i)$$
$$s_{i+1} = a_i \oplus b_i \oplus c_i$$

Gatterdarstellung

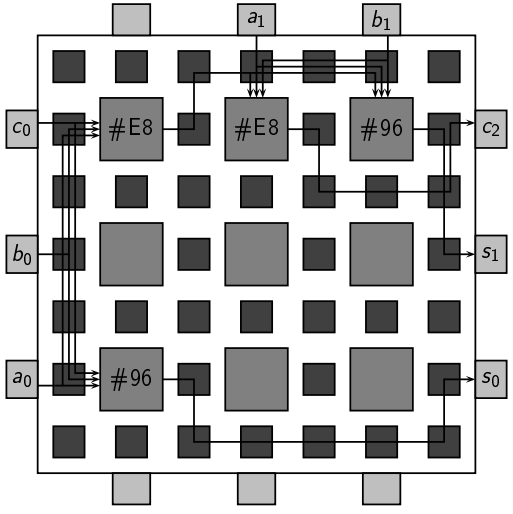


LUT-Graph (Mapping)

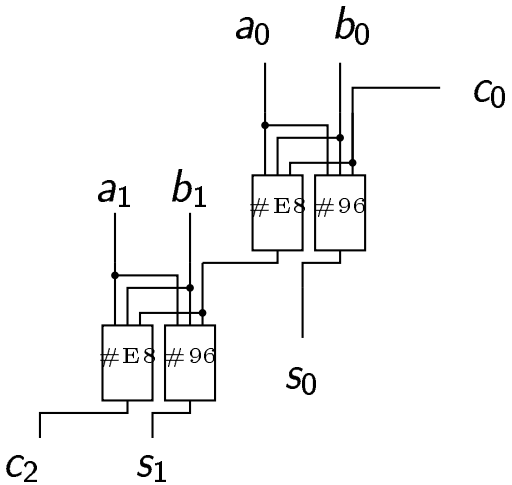
LUT-Graph (Mapping)



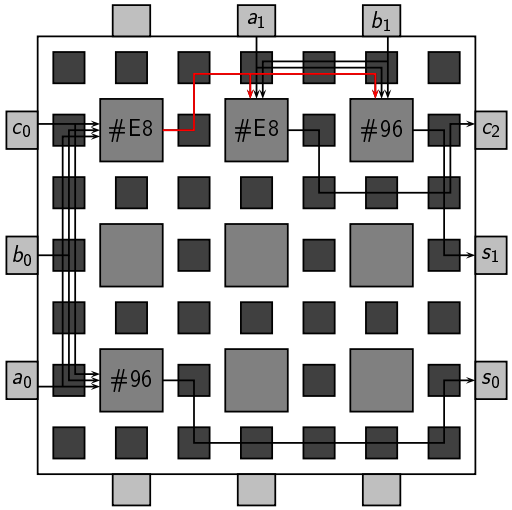
FPGA (Place&Route)



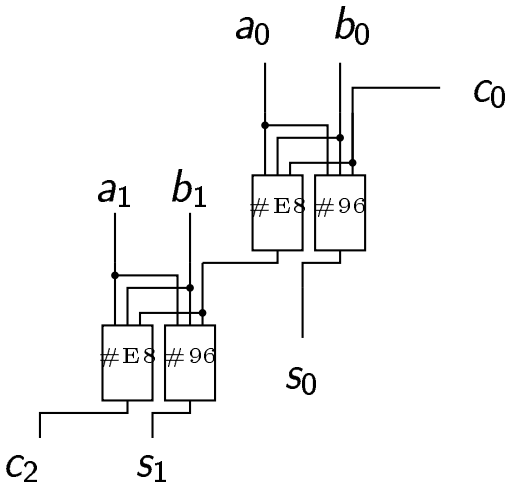
LUT-Graph (Mapping)



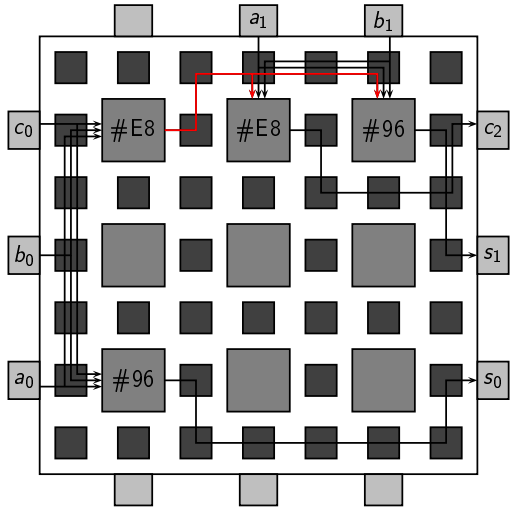
FPGA (Place&Route)



LUT-Graph (Mapping)



FPGA (Place&Route)

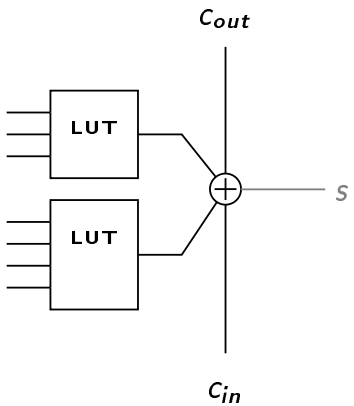


Verbesserung: Umgehung der Routing-Matrix durch Benutzung der Carry-Logik

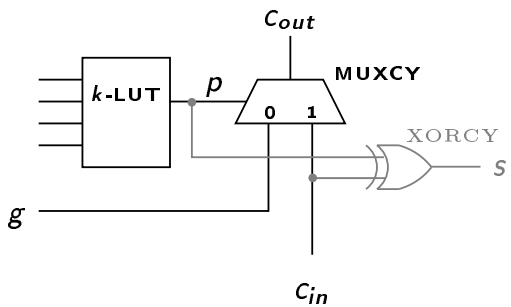
Carry-Chain-Strukturen

Definition

Eine lineare Hintereinanderschaltung von Carry-Logik in Logikblöcken heißt (Hardware-)Carry-Chain.

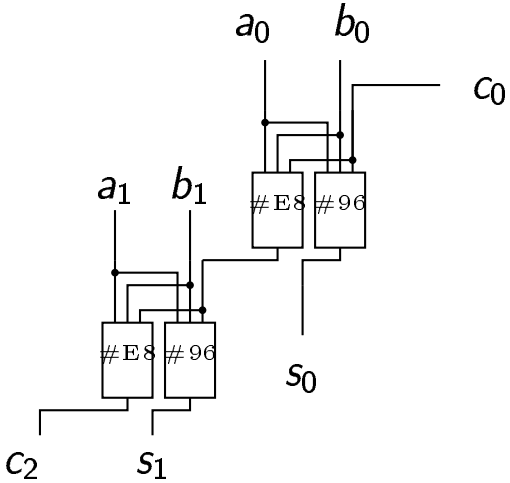


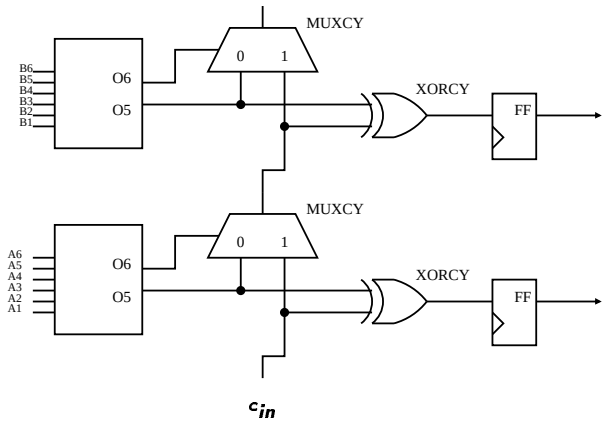
Stratix II
Ripple-Carry-Addierer



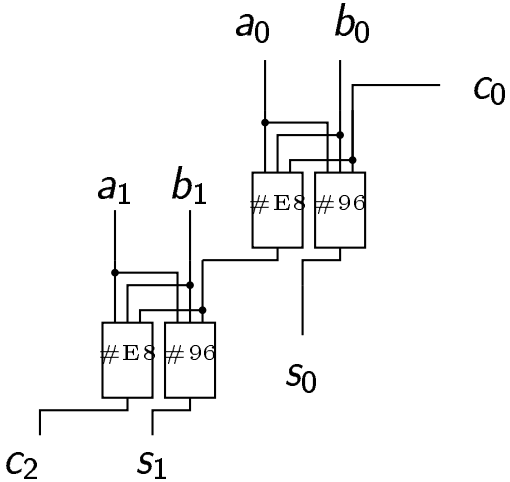
Spartan/Virtex
Manchester-Carry-Chain

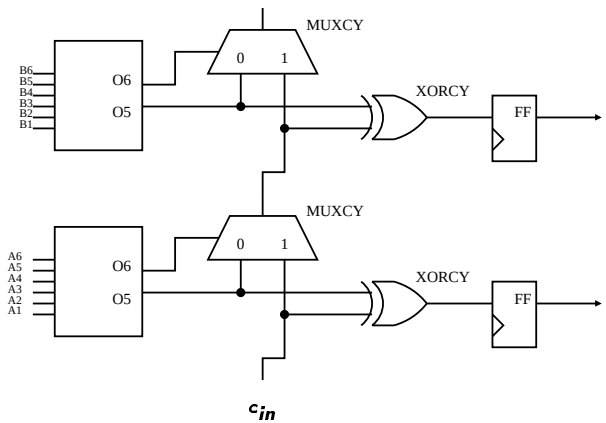
Frage: Wie kann festgestellt werden, ob Teile einer Schaltung auf Carry-Chains abbildbar sind?





Frage: Wie kann festgestellt werden, ob Teile einer Schaltung auf Carry-Chains abbildbar sind?

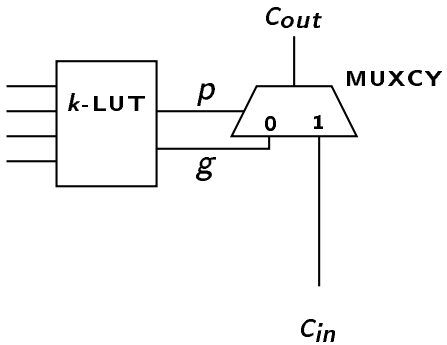




Antwort: Untersuche die verwendeten Funktionen im LUT-Graphen auf Abbildbarkeit auf Grundzellen.

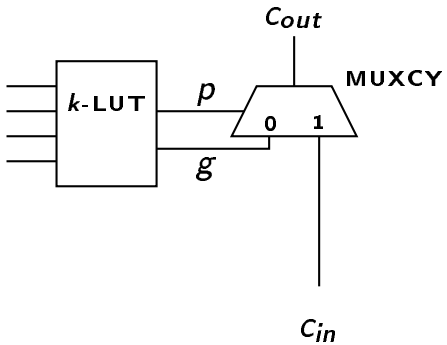
Funktionsgleichung des Multiplexers

$$
\begin{aligned}
c_{out} &= \begin{cases} g, & \text{wenn } p = 0 \\ c_{in}, & \text{wenn } p = 1 \end{cases} \\
&= \overline{p}g + pc_{in}
\end{aligned}
$$



Funktionsgleichung des Multiplexers

$$c_{out} = \begin{cases} g, & \text{wenn } p = 0 \\ c_{in}, & \text{wenn } p = 1 \end{cases} = \overline{p}g + pc_{in}$$



Definition

Eine Funktion $F : \{0,1\}^{k+1} \to \{0,1\}$ heißt **Xilinx-Carry-Chain-nutzbar**, wenn es für ein x_i Funktionen $P_i, G_i : \mathbb{B}^k \to \{0,1\}$ gibt, so dass

$$F(x_i, \underline{x}) = \overline{P_i(\underline{x})} G_i(\underline{x}) + x_i P_i(\underline{x}).$$

Charakterisierung von XCC-nutzbaren Funktionen

Lemma (SHANNON)

Sei $F : \{0,1\}^{k+1} \to \{0,1\}$ *eine binäre Funktion. Dann gibt es für alle* $i \in \{0,\ldots,k\}$ *binäre Funktionen* $F_i^0, F_i^1 : \{0,1\}^k \to \{0,1\}$ *mit*

$$F(x_0,\ldots,x_{k-1}) \quad = \quad \overline{x_i} F_i^0 + x_i F_i^1$$

und es gilt $F_i^0 = F\big|_{x_i=0}$, $F_i^1 = F\big|_{x_i=1}$.

Lemma

Eine Funktion F ist XCC-nutzbar genau dann, wenn es eine SHANNON-Zerlegung $F(c_{in}, \underline{x}) = \overline{c_{in}} F_0(\underline{x}) + c_{in} F_1(\underline{x})$ gibt, für deren Kofaktoren gilt, dass

$$F_1(\underline{x}) \Rightarrow F_0(\underline{x}).$$

Lemma

Eine Funktion F ist XCC-nutzbar genau dann, wenn es eine SHANNON-Zerlegung $F(c_{in}, \underline{x}) = \overline{c_{in}} F_0(\underline{x}) + c_{in} F_1(\underline{x})$ gibt, für deren Kofaktoren gilt, dass

$$F_1(\underline{x}) \Rightarrow F_0(\underline{x}).$$

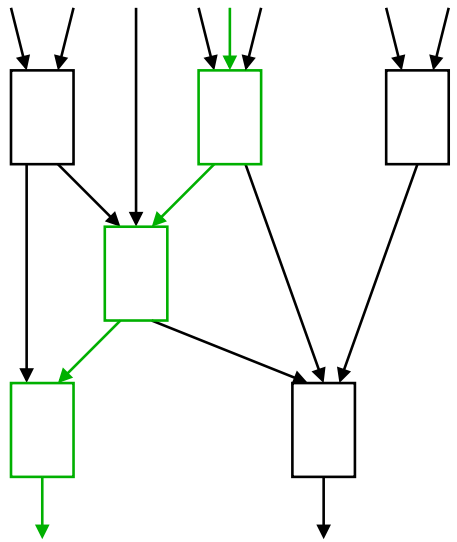
Lemma

Ist eine Funktion F XCC-nutzbar mit Carry-Signal c_{in}, dann können die Funktionen P und G durch

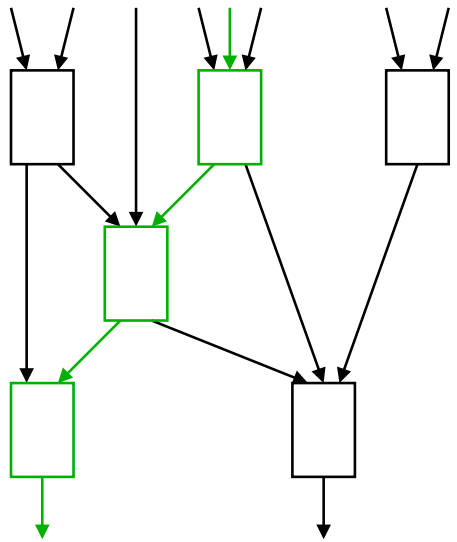
$$P = \overline{F_0} F_1 \qquad \text{und} \qquad G \in \{F_1, F_1 F_0\}$$

ermittelt werden.

Ansatz 1: Suchen einer längsten Carry-Chain, bzw. Suchen eines längsten Pfades im LUT-Graphen mit ausschließlich nutzbaren Knoten



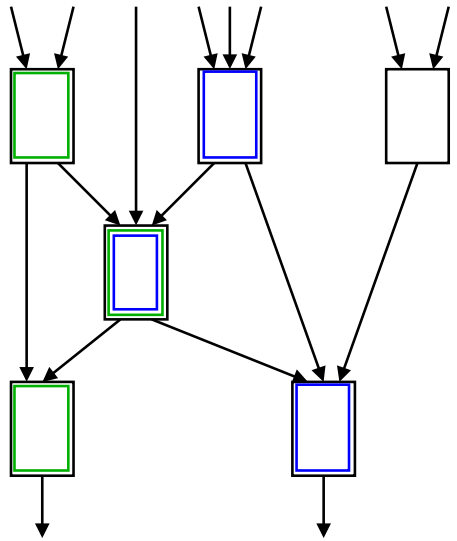
Ansatz 1: Suchen einer längsten Carry-Chain, bzw. Suchen eines längsten Pfades im LUT-Graphen mit ausschließlich nutzbaren Knoten



Vorteile: bekannte Algorithmen existent

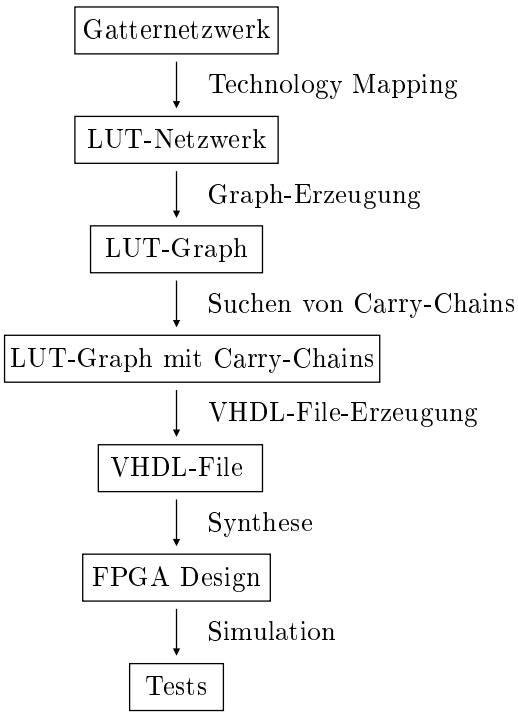
Nachteile: Überdecken des kritischen Pfades unwahrscheinlich

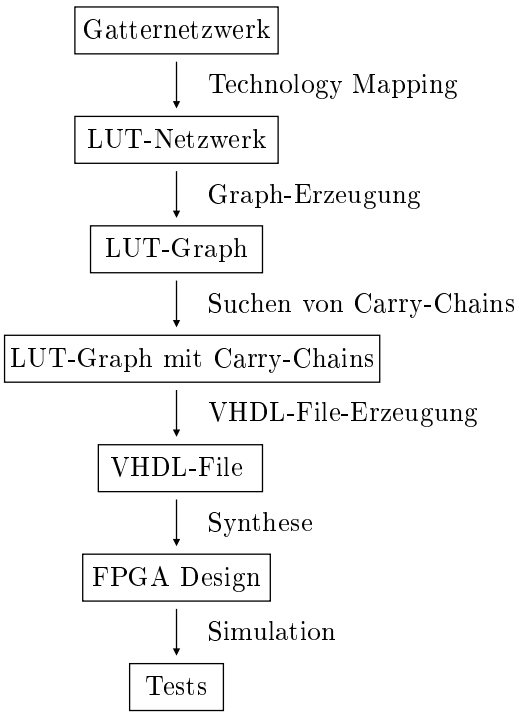
Ansatz 2: Mehrfaches Anwenden der Methode "Suchen der längsten Carry-Chain"

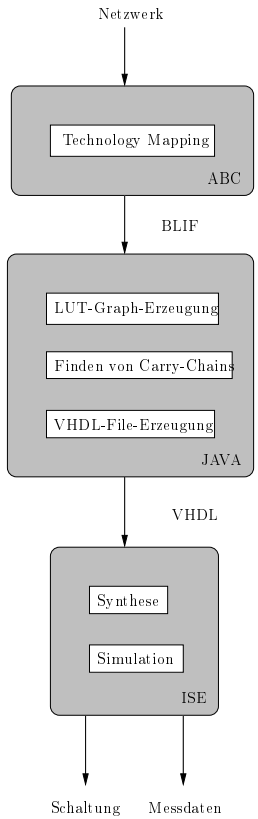


Nachteile: Ketteneigenschaft muss erhalten bleiben

- Kopien erzeugen
- Vermeidung von bereits benutzten Knoten

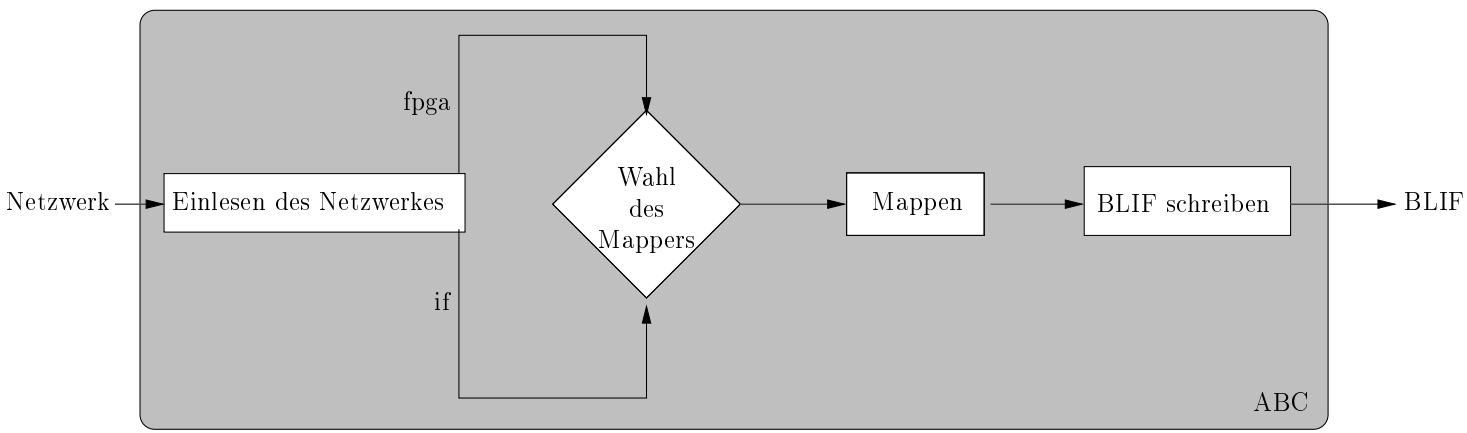

Gatternetzwerk
Technology Mapping
LUT-Netzwerk
Graph-Erzeugung
LUT-Graph
Suchen von Carry-Chains
LUT-Graph mit Carry-Chains
VHDL-File-Erzeugung
VHDL-File
Synthese
FPGA Design
Simulation
Tests


Gatternetzwerk
Technology Mapping
LUT-Netzwerk
Graph-Erzeugung
LUT-Graph
Suchen von Carry-Chains
LUT-Graph mit Carry-Chains
VHDL-File-Erzeugung
VHDL-File
Synthese
FPGA Design
Simulation
Tests

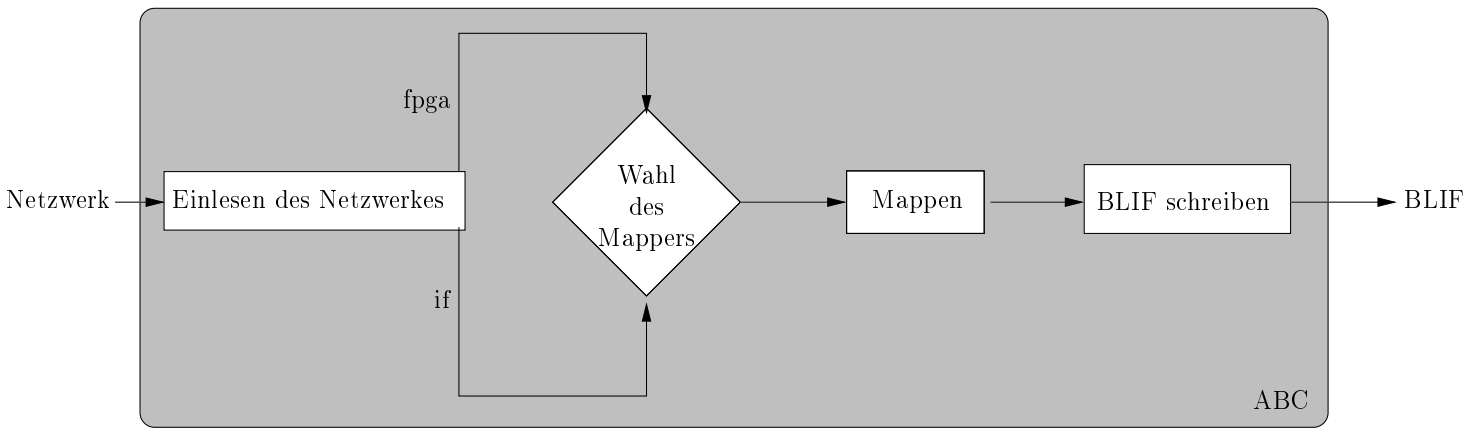

Netzwerk
Technology Mapping
ABC
BLIF
LUT-Graph-Erzeugung
Finden von Carry-Chains
VHDL-File-Erzeugung
JAVA
VHDL
Synthese
Simulation
ISE
Schaltung
Messdaten

- Steuerung des Mapping-Prozesses durch ABC

- Steuerung des Mapping-Prozesses durch ABC
- Wahl zwischen den Mapping-Funktionen if, ffpga möglich

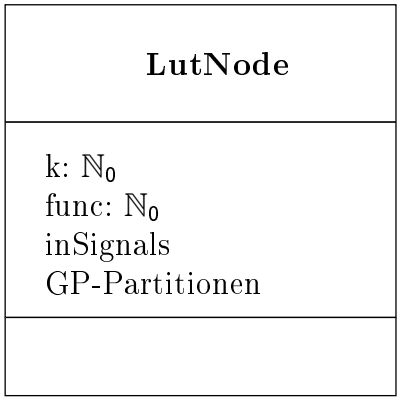


- Steuerung des Mapping-Prozesses durch ABC
- Wahl zwischen den Mapping-Funktionen if, ffpga möglich
- Ergebnis ist ein BLIF



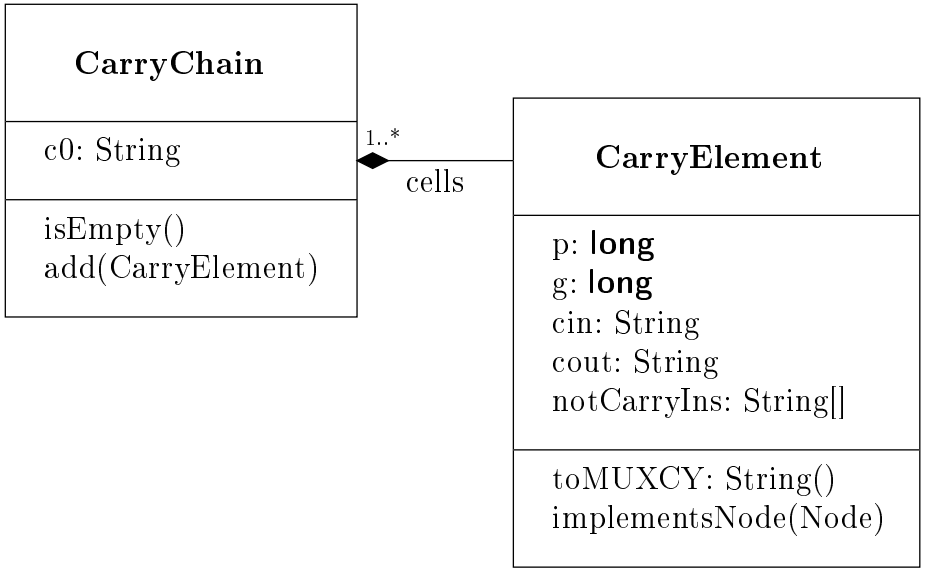
- interne Darstellung über Mengen von Knoten und Kanten (JGraphT)
- Definition spezifischer Knotentypen: InputNode, OutputNode, LutNode

- interne Darstellung über Mengen von Knoten und Kanten (JGraphT)
- Definition spezifischer Knotentypen: InputNode, OutputNode, LutNode

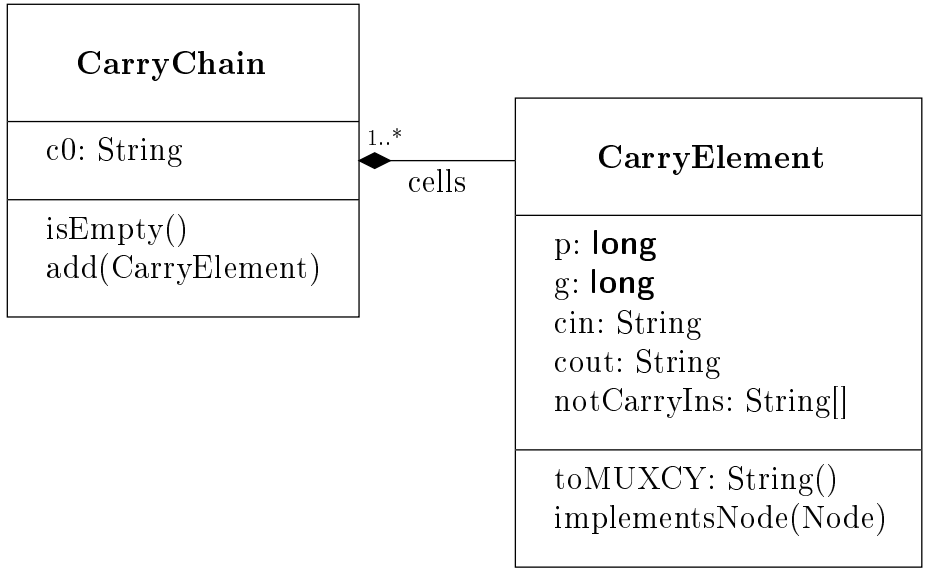


- Zur Speicherung wurde ein Datentyp CarryChain erzeugt

- Zur Speicherung wurde ein Datentyp CarryChain erzeugt

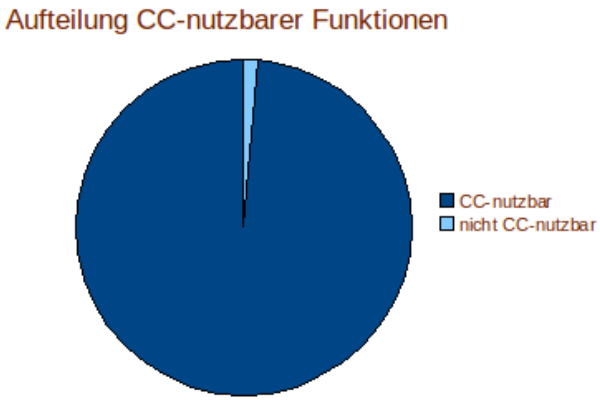


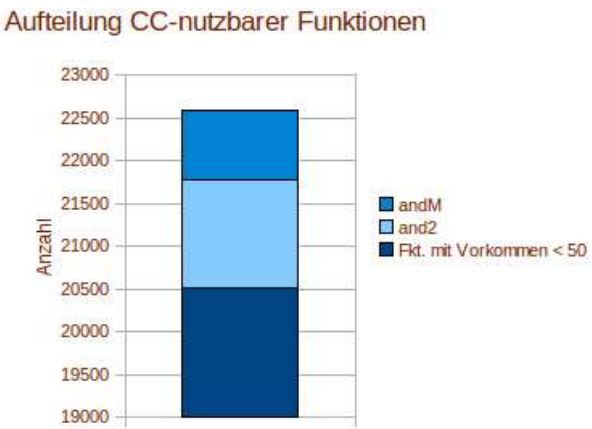
- Zur Speicherung wurde ein Datentyp CarryChain erzeugt

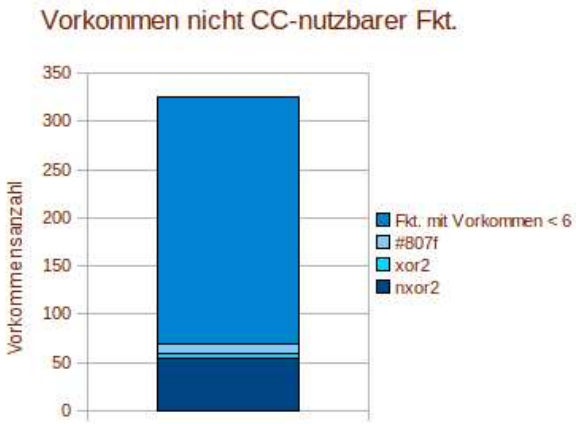


- Erzeugung von Carry-Chains durch einen invertierten Floyd-Warshall-Algorithmus (Methode längste Carry-Chain)

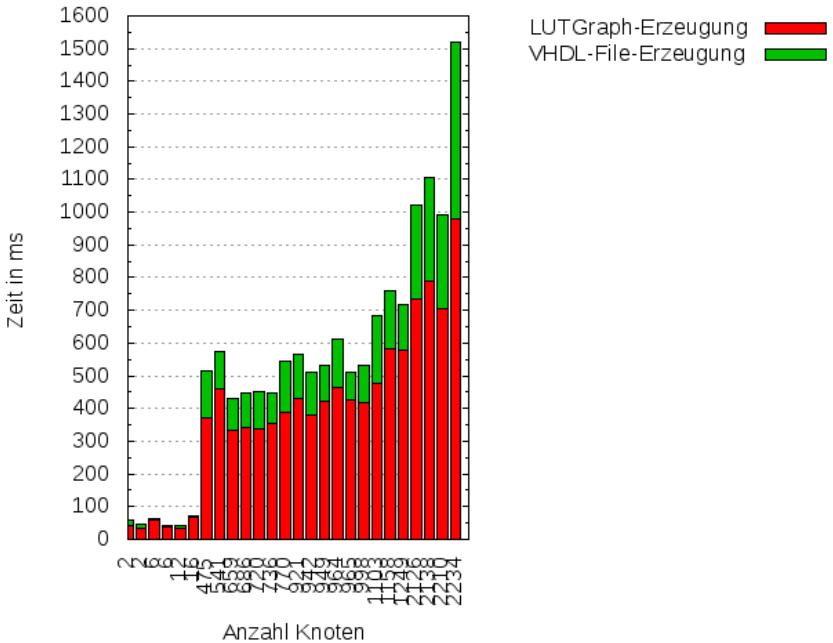
Nutzbare Funktionen der MCNC-Benchmarks

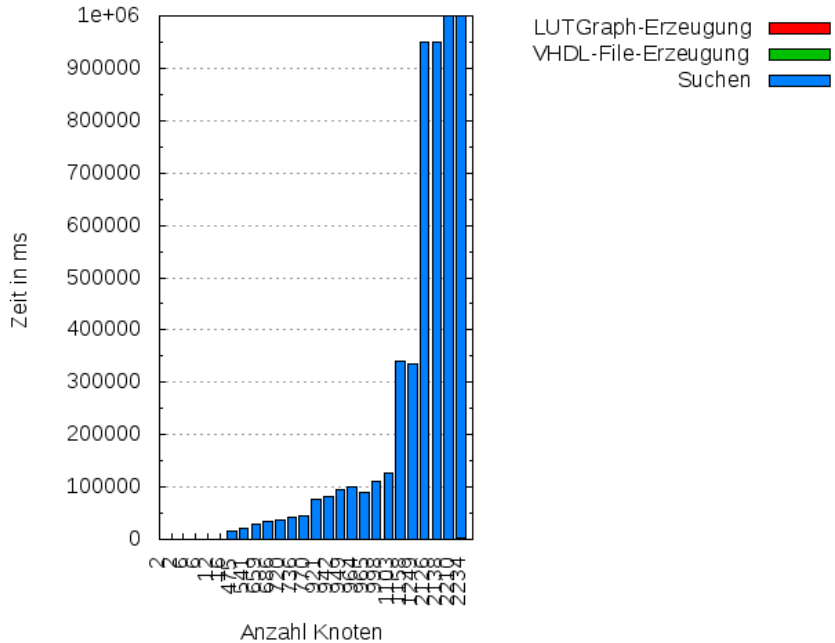




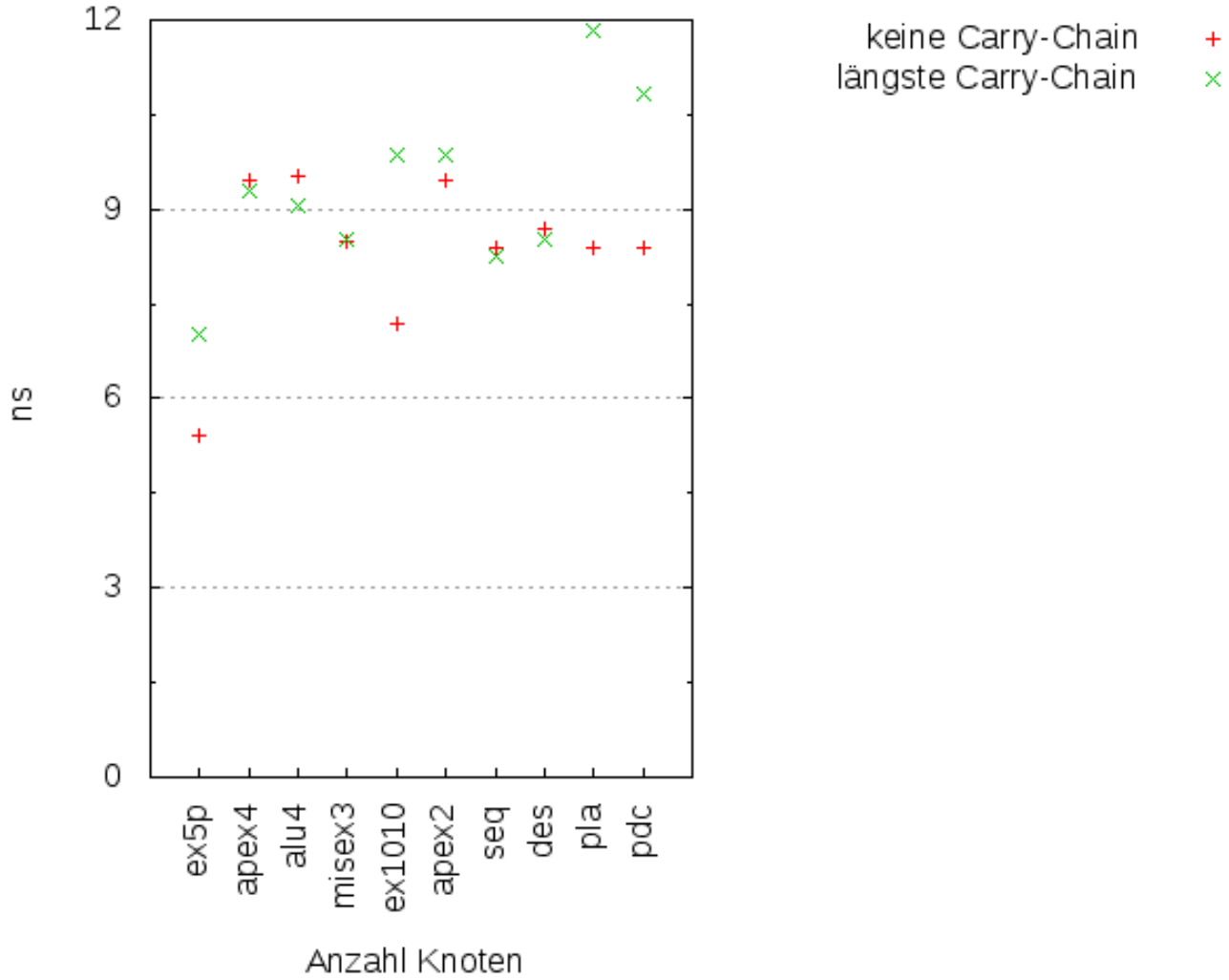


Laufzeiten


LUTGraph-Erzeugung
VHDL-File-Erzeugung
Zeit in ms
Anzahl Knoten


LUTGraph-Erzeugung
VHDL-File-Erzeugung
Suchen
Zeit in ms
Anzahl Knoten

Verzögerungszeiten für MCNC-Testbench



Verbesserungen und Aussicht

- Bestimmung von P und G mittels Cashing

Verbesserungen und Aussicht

- Bestimmung von P und G mittels Cashing
- weitere Suchalgorithmen für Carry-Chains

Verbesserungen und Aussicht

- Bestimmung von P und G mittels Cashing
- weitere Suchalgorithmen für Carry-Chains
- Kombination von Carry-Chains mit spezifischen Bauteilen auf einem FPGA

Verbesserungen und Aussicht

- Bestimmung von P und G mittels Cashing
- weitere Suchalgorithmen für Carry-Chains
- Kombination von Carry-Chains mit spezifischen Bauteilen auf einem FPGA
- Untersuchung von Verzögerungsmaßen zur Bestimmung kritischer Pfade

Verbesserungen und Aussicht

- Bestimmung von P und G mittels Cashing
- weitere Suchalgorithmen für Carry-Chains
- Kombination von Carry-Chains mit spezifischen Bauteilen auf einem FPGA
- Untersuchung von Verzögerungsmaßen zur Bestimmung kritischer Pfade
- Auffassung des LUT-Graphen als Ordnungsdiagramm. Können Erkenntnisse und Verbesserungen durch Linearisierung (z. B. Algorithmus von LAWLER) gewonnen werden?

Vielen Dank für Ihre Aufmerksamkeit