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

Julia Daniel

16. 12. 2009

Motivation

Mapping

Ausblick

Motivation

Für einen allgemeinen Schaltkreis ist

Verzögerungszeit	$\approx$ 70 % Routing-Verzögerung
	Großteil des Restes ist LUT-Verzögerung

Motivation

Für einen allgemeinen Schaltkreis ist

Verzögerungszeit	$\approx$ 70 % Routing-Verzögerung
	Großteil des Restes ist LUT-Verzögerung

Ziel:

- Vermeidung der Routing-Matrix durch Carry-Chains
- Formulierung eines Mapping-Algorithmuses, der zwischen Routing- und Logik-Tiefe unterscheidet

Mapping

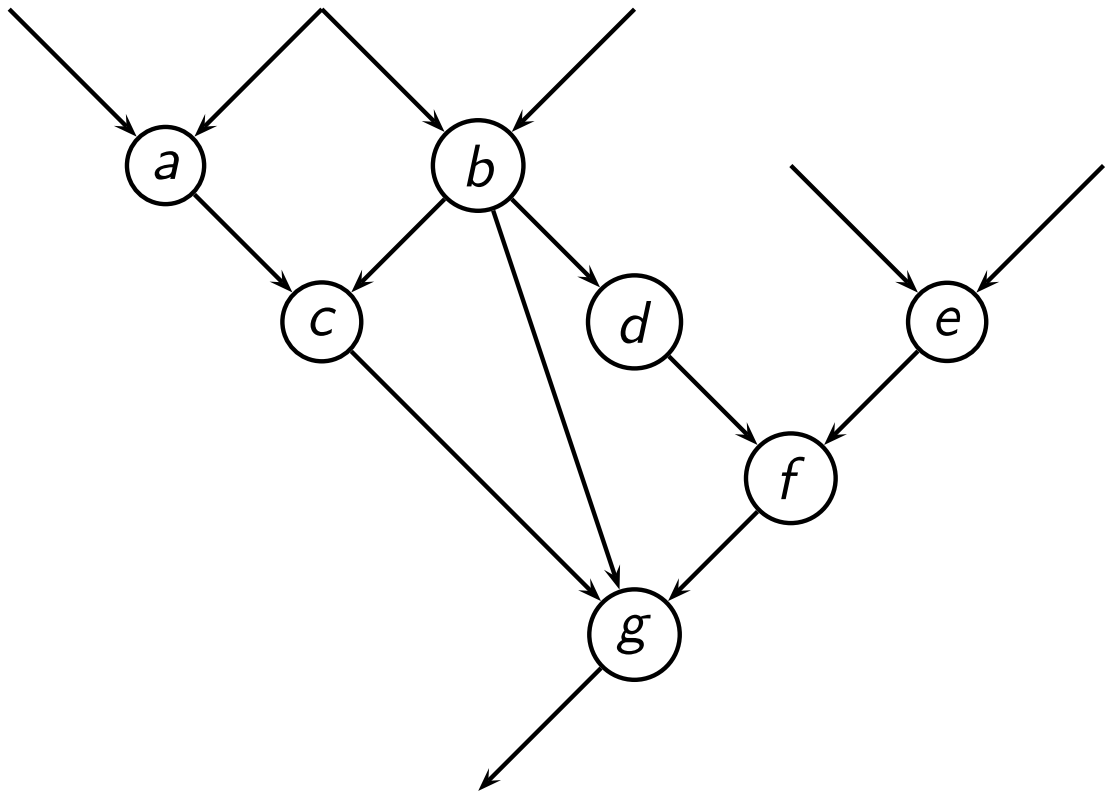
Gegeben: DAG $N = (V, E)$ und ein FPGA mit K-LUTs

Mapping

Gegeben: DAG $N = (V, E)$ und ein FPGA mit K-LUTs
Ziel: überdecke N mit Kegeln der Breite K (K-LUTs)

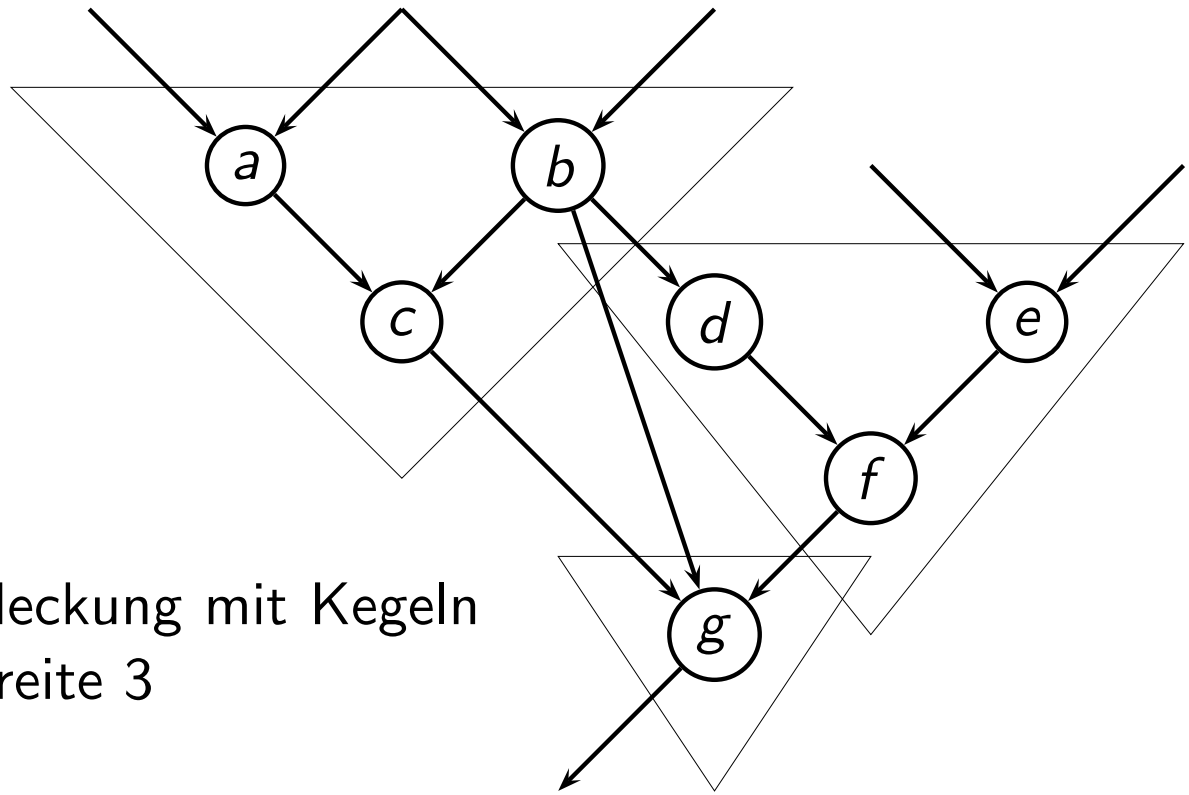
Mapping

Gegeben: DAG $N = (V, E)$ und ein FPGA mit K-LUTs
Ziel: überdecke N mit Kegeln der Breite K (K-LUTs)



Mapping

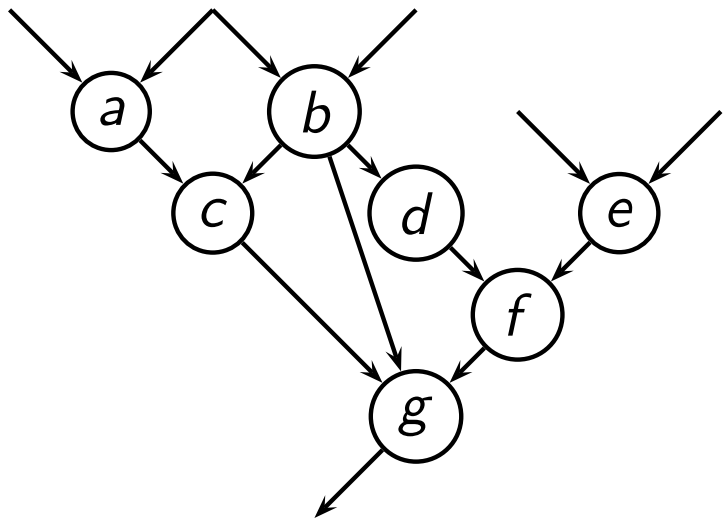
Gegeben: DAG $N = (V, E)$ und ein FPGA mit K-LUTs
Ziel: überdecke N mit Kegeln der Breite K (K-LUTs)



Überdeckung mit Kegeln der Breite 3

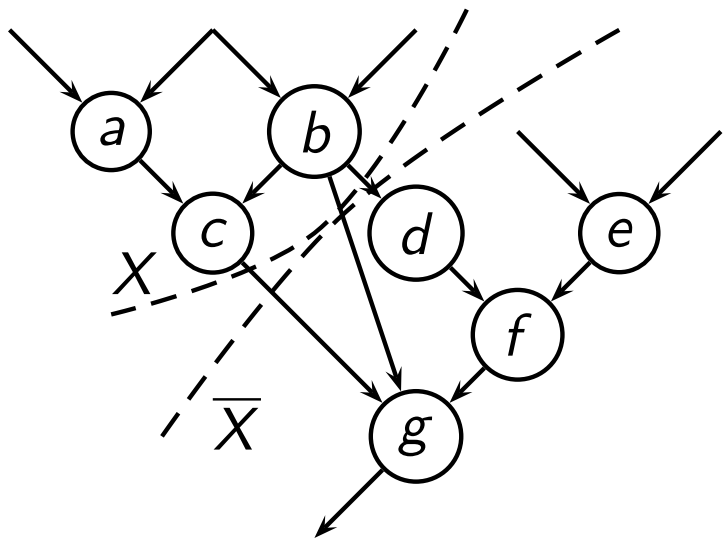
Schnitte in DAGs

Für einen Knoten $t \in N$ ist ein Schnitt $\left(X_t, \overline{X}_t\right)$ eine Partitionierung von N, so daß t und seine Nachfolger in $\overline{X}_t$ sind Beispiel



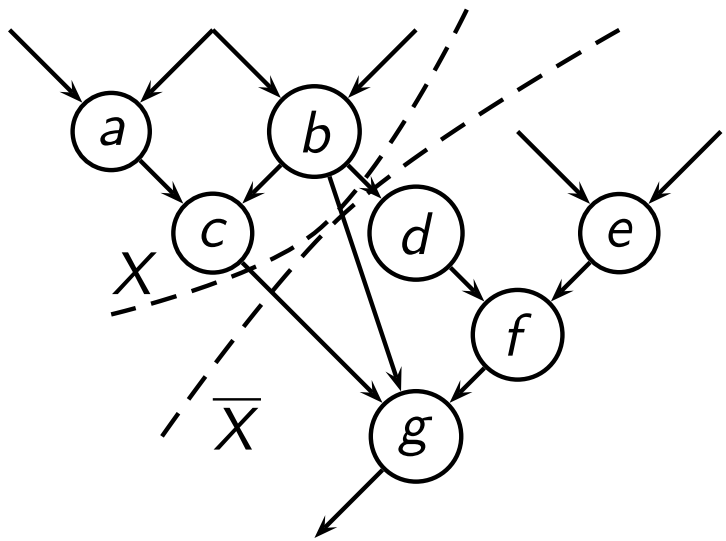
Schnitte in DAGs

Für einen Knoten $t \in N$ ist ein Schnitt $\left(X_t, \overline{X}_t\right)$ eine Partitionierung von N, so daß t und seine Nachfolger in $\overline{X}_t$ sind Beispiel



Schnitte in DAGs

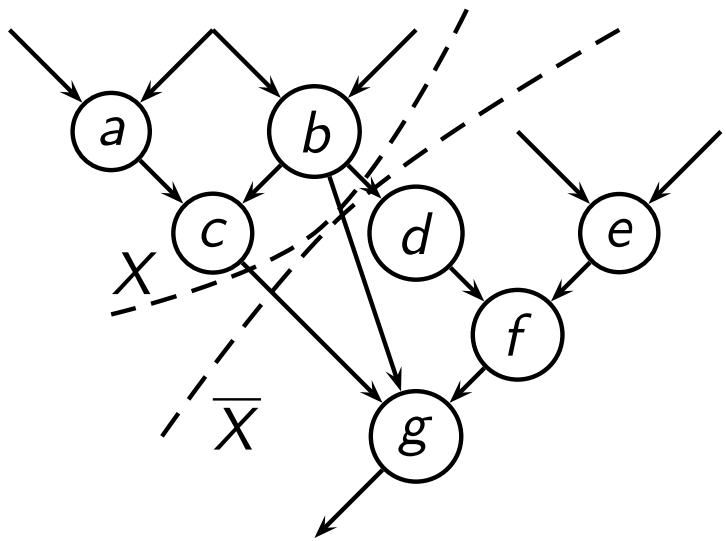
Für einen Knoten $t \in N$ ist ein Schnitt $(X_t, \overline{X}_t)$ eine Partitionierung von N, so daß t und seine Nachfolger in $\overline{X}_t$ sind
Beispiel



$$inputs\,(X, \overline{X}) = \{b, c\}$$

Schnitte in DAGs

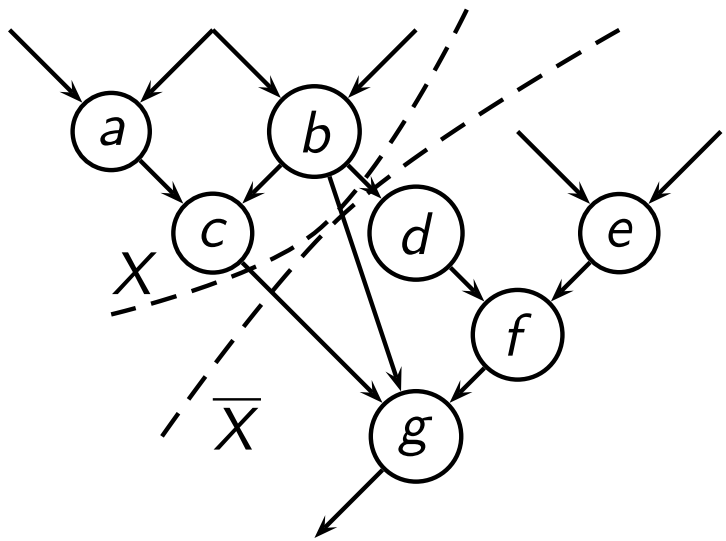
Für einen Knoten $t \in N$ ist ein Schnitt $(X_t, \overline{X}_t)$ eine Partitionierung von N, so daß t und seine Nachfolger in $\overline{X}_t$ sind
Beispiel



$$inputs\,(X, \overline{X}) = \{b, c\}$$
$$\Longrightarrow \; (X, \overline{X}) \text{ ist 2-beschränkt}$$

Schnitte in DAGs

Für einen Knoten $t \in N$ ist ein Schnitt $\left(X_t, \overline{X}_t\right)$ eine Partitionierung von N, so daß t und seine Nachfolger in $\overline{X}_t$ sind
Beispiel



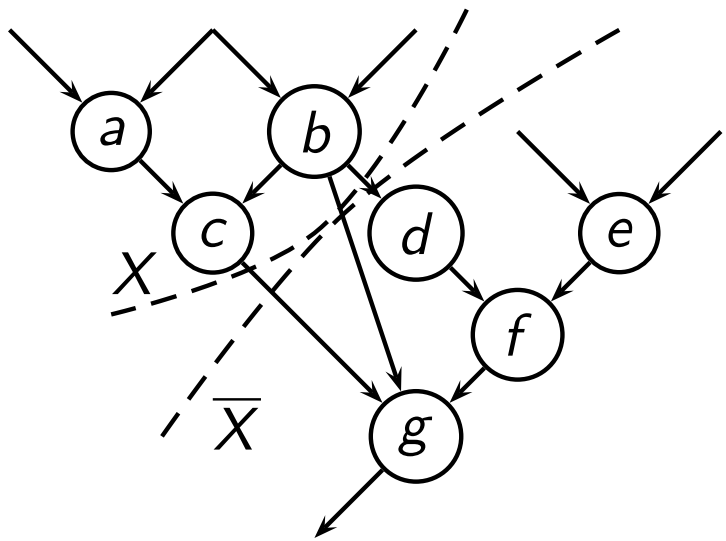
$$inputs\left(X, \overline{X}\right) = \{b, c\}$$

$$\Longrightarrow \left(X, \overline{X}\right) \text{ ist 2-beschränkt}$$

$$h_{level}\left(X, \overline{X}\right) = 2$$

Schnitte in DAGs

Für einen Knoten $t \in N$ ist ein Schnitt $\left(X_t, \overline{X}_t\right)$ eine Partitionierung von N, so daß t und seine Nachfolger in $\overline{X}_t$ sind
Beispiel



$$inputs\left(X, \overline{X}\right) = \{b, c\}$$

$$\Longrightarrow \left(X, \overline{X}\right) \text{ ist 2-beschränkt}$$

$$h_{level}\left(X, \overline{X}\right) = 2$$

Feststellung: Finden aller K-LUTs ist äquivalent zum Finden aller K-beschränkten Schnitte

ChainMap

ChainMap[1]

- ist ein polynom-Zeit Mapping-Algorithmus

[1] Michael T. Frederick, Arun K. Somani: Beyond the Arithmetic Constraint: Depth-Optimal Mapping of Logic Chains on LUT-based FPGAs

ChainMap

ChainMap[1]

- ist ein polynom-Zeit Mapping-Algorithmus
- basiert auf Schnitten in Graphen

[1] Michael T. Frederick, Arun K. Somani: Beyond the Arithmetic Constraint: Depth-Optimal Mapping of Logic Chains on LUT-based FPGAs

ChainMap

ChainMap[1]

- ist ein polynom-Zeit Mapping-Algorithmus
- basiert auf Schnitten in Graphen
- optimiert Routing-Tiefe

[1] Michael T. Frederick, Arun K. Somani: Beyond the Arithmetic Constraint: Depth-Optimal Mapping of Logic Chains on LUT-based FPGAs

ChainMap

ChainMap[1]

- ist ein polynom-Zeit Mapping-Algorithmus
- basiert auf Schnitten in Graphen
- optimiert Routing-Tiefe
- erzeugt logische Ketten (Carry-Chains)

[1] Michael T. Frederick, Arun K. Somani: Beyond the Arithmetic Constraint: Depth-Optimal Mapping of Logic Chains on LUT-based FPGAs

ChainMap

ChainMap[1]

- ist ein polynom-Zeit Mapping-Algorithmus
- basiert auf Schnitten in Graphen
- optimiert Routing-Tiefe
- erzeugt logische Ketten (Carry-Chains)
- funktioniert nur für FPGAs mit spezieller LUT-Struktur, insbesondere nicht für Stratix-II und Xilinx

[1] Michael T. Frederick, Arun K. Somani: Beyond the Arithmetic Constraint: Depth-Optimal Mapping of Logic Chains on LUT-based FPGAs

ChainMap

Phasen

1. Labeling: Beschriftung der Knoten $t \in N$ mit Routing-Tiefe $g(t)$ und Logik-Tiefe $l(t)$, Finden eines minimalen Schnittes

ChainMap

Phasen

1. Labeling: Beschriftung der Knoten $t \in N$ mit Routing-Tiefe $g(t)$ und Logik-Tiefe $l(t)$, Finden eines minimalen Schnittes
2. Mapping

ChainMap

Phasen

1. Labeling: Beschriftung der Knoten $t \in N$ mit Routing-Tiefe $g(t)$ und Logik-Tiefe $l(t)$, Finden eines minimalen Schnittes
2. Mapping
3. Duplizieren: Herstellen der Ketteneigenschaft

ChainMap

Labeling

- für alle $t \in \mathrm{PI}$ setze $g(t) = l(t) = 0$

ChainMap

Labeling

- für alle $t \in \mathrm{PI}$ setze $g(t) = l(t) = 0$
- für alle $t \in N \backslash \mathrm{PI}$:

ChainMap

Labeling

- für alle $t \in \mathrm{PI}$ setze $g(t) = l(t) = 0$
- für alle $t \in N \backslash \mathrm{PI}$:
  - bestimme den Vorgängergraphen N_t

ChainMap

Labeling

- für alle $t \in \mathrm{PI}$ setze $g(t) = l(t) = 0$
- für alle $t \in N \backslash \mathrm{PI}$:
  - bestimme den Vorgängergraphen N_t
  - bestimme $p = \max\{g(u) : u \in \mathit{inputs}(t)\}$

ChainMap

Labeling

- für alle $t \in \mathrm{PI}$ setze $g(t) = l(t) = 0$
- für alle $t \in N \backslash \mathrm{PI}$:
  - bestimme den Vorgängergraphen N_t
  - bestimme $p = \max\{g(u) : u \in \mathit{inputs}(t)\}$
  - bestimme $q = \max\{l(u) : u \in \mathit{inputs}(t)\}$

ChainMap

Labeling

- für alle $t \in \mathrm{PI}$ setze $g(t) = l(t) = 0$
- für alle $t \in N \backslash \mathrm{PI}$:
  - bestimme den Vorgängergraphen N_t
  - bestimme $p = \max\{g(u) : u \in \mathit{inputs}(t)\}$
  - bestimme $q = \max\{l(u) : u \in \mathit{inputs}(t)\}$
  - $P := \{u \in N_t : g(u) = p\}$

ChainMap

Labeling

- für alle $t \in \mathrm{PI}$ setze $g(t) = l(t) = 0$
- für alle $t \in N \backslash \mathrm{PI}$:
  - bestimme den Vorgängergraphen N_t
  - bestimme $p = \max\{g(u) : u \in \mathit{inputs}(t)\}$
  - bestimme $q = \max\{l(u) : u \in \mathit{inputs}(t)\}$
  - $P := \{u \in N_t : g(u) = p\}$
  - überprüfe, ob es einen tiefen-erhöhenden Knoten gibt

ChainMap

Labeling

- für alle $t \in \mathrm{PI}$ setze $g(t) = l(t) = 0$
- für alle $t \in N \backslash \mathrm{PI}$:
  - bestimme den Vorgängergraphen N_t
  - bestimme $p = \max\{g(u) : u \in \mathit{inputs}(t)\}$
  - bestimme $q = \max\{l(u) : u \in \mathit{inputs}(t)\}$
  - $P := \{u \in N_t : g(u) = p\}$
  - überprüfe, ob es einen tiefen-erhöhenden Knoten gibt
    - ja: setze $g(t) = p + 1$, $l(t) = q + 1$, Schnitt$= (N_t, \emptyset)$

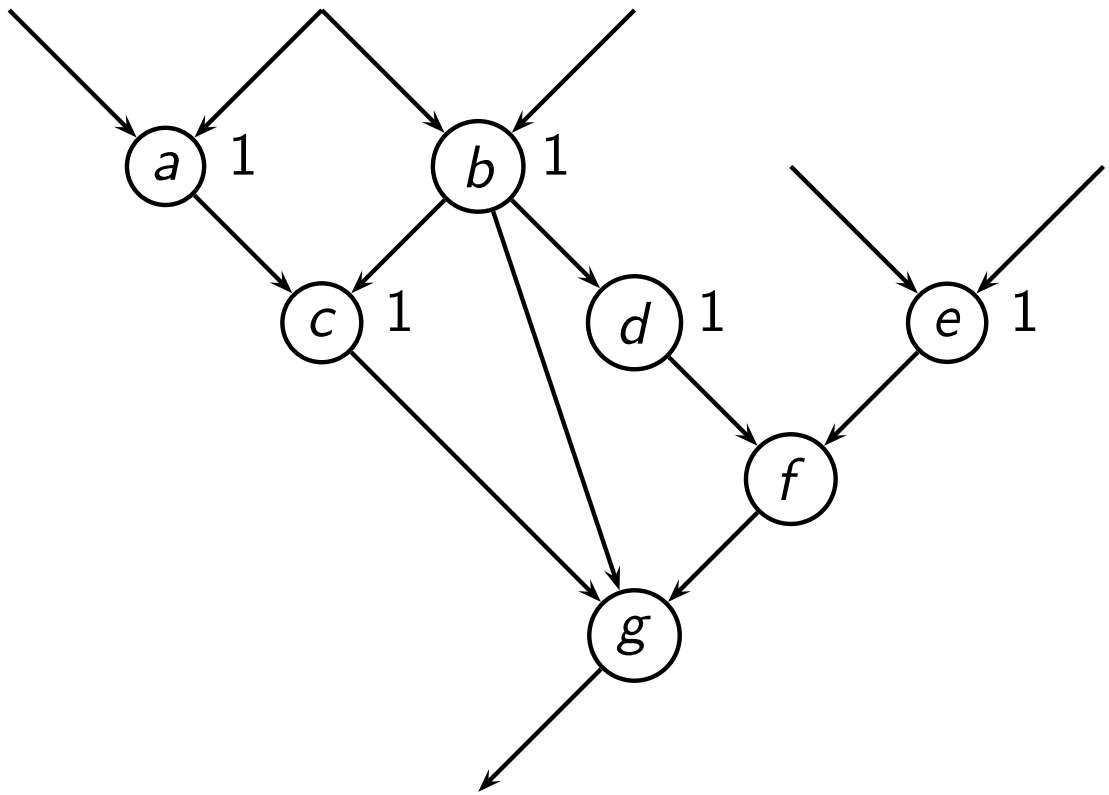
ChainMap

Labeling

- für alle $t \in \mathrm{PI}$ setze $g(t) = l(t) = 0$
- für alle $t \in N \backslash \mathrm{PI}$:
  - bestimme den Vorgängergraphen N_t
  - bestimme $p = \max\{g(u) : u \in inputs(t)\}$
  - bestimme $q = \max\{l(u) : u \in inputs(t)\}$
  - $P := \{u \in N_t : g(u) = p\}$
  - überprüfe, ob es einen tiefen-erhöhenden Knoten gibt
    - ja: setze $g(t) = p + 1$, $l(t) = q + 1$, Schnitt$= (N_t, \emptyset)$
    - nein: isetze $g(t) = p$, $l(t) = q + 1$ modifiziere N_t so, daß ein minimaler Schnitt durch das Problem der maximalen Flüsse in Graphen gelöst werden kann

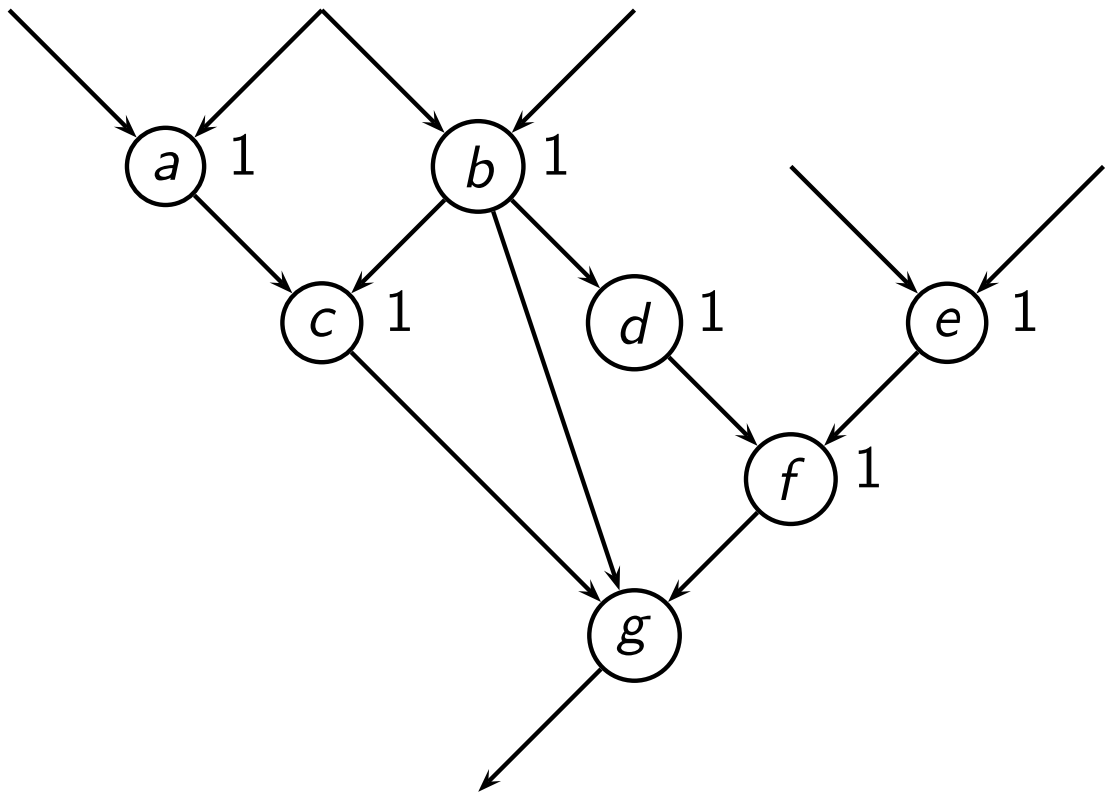
ChainMap

Labeling



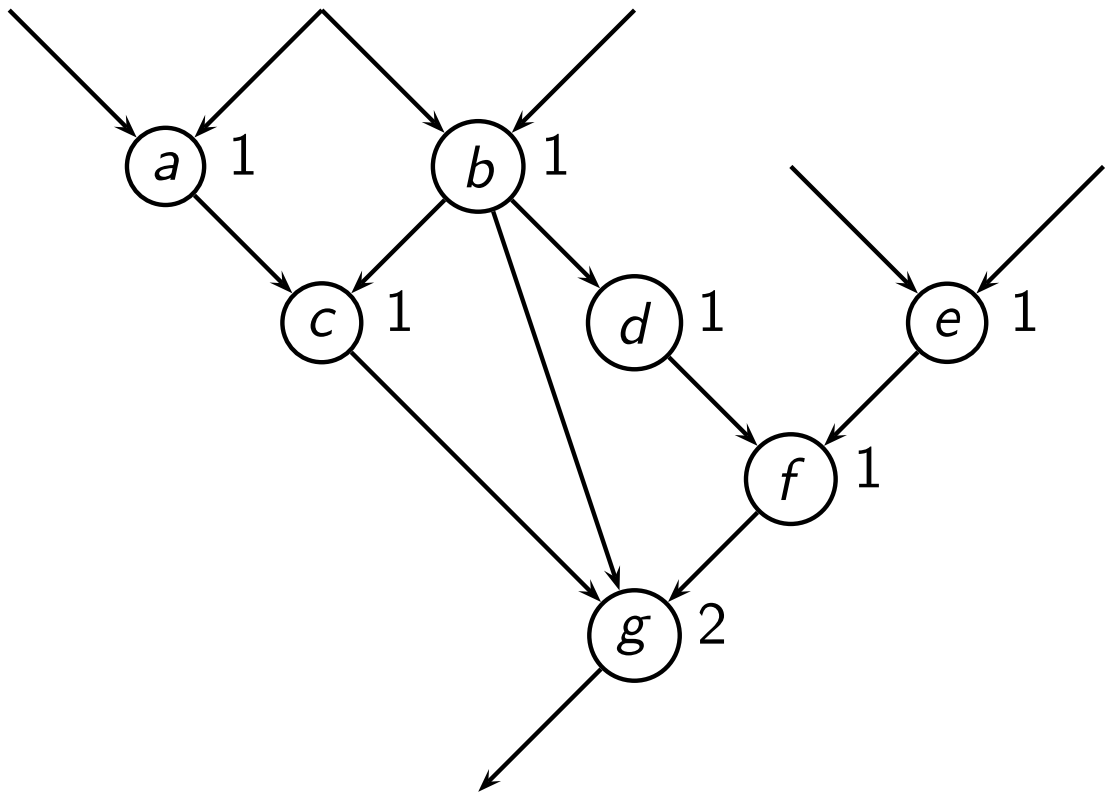
ChainMap

Labeling



ChainMap

Labeling



ChainMap

Mapping

- $T := \{PO\}$

ChainMap

Mapping

- $T := \{PO\}$
- berechne für $t \in T$ die K-LUT t' solange, bis T nur noch PIs enthält:

ChainMap

Mapping

- $T := \{PO\}$
- berechne für $t \in T$ die K-LUT t' solange, bis T nur noch PIs enthält:
  - nehme einen minimalen K-beschränkten Schnitt $\left(X_t, \overline{X}_t\right)$ aus der 1. Phase

ChainMap

Mapping

- $T := \{PO\}$
- berechne für $t \in T$ die K-LUT t' solange, bis T nur noch PIs enthält:
  - nehme einen minimalen K-beschränkten Schnitt $\left(X_t, \overline{X}_t\right)$ aus der 1. Phase
  - setze $\mathit{inputs}\left(t'\right) := \mathit{inputs}\left(\overline{X}_t\right)$

ChainMap

Mapping

- $T := \{PO\}$
- berechne für $t \in T$ die K-LUT t' solange, bis T nur noch PIs enthält:
  - nehme einen minimalen K-beschränkten Schnitt $\left(X_t, \overline{X}_t\right)$ aus der 1. Phase
  - setze $\mathit{inputs}\left(t'\right) := \mathit{inputs}\left(\overline{X}_t\right)$
  - beschreibe die Funktion von t' durch die Knoten in $\overline{X}_t$

ChainMap

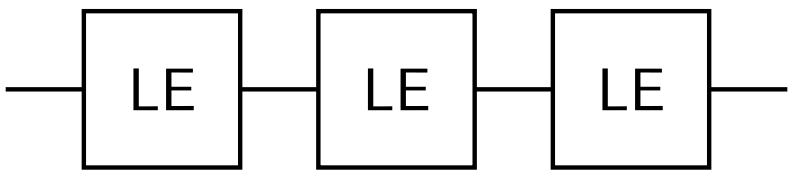
Mapping

- $T := \{PO\}$
- berechne für $t \in T$ die K-LUT t' solange, bis T nur noch PIs enthält:
  - nehme einen minimalen K-beschränkten Schnitt $\left(X_t, \overline{X}_t\right)$ aus der 1. Phase
  - setze $\mathit{inputs}\left(t'\right) := \mathit{inputs}\left(\overline{X}_t\right)$
  - beschreibe die Funktion von t' durch die Knoten in $\overline{X}_t$
  - $T := T \cup \mathit{inputs}\left(\overline{X}_t\right) \setminus \{t\}$

ChainMap

Duplikation

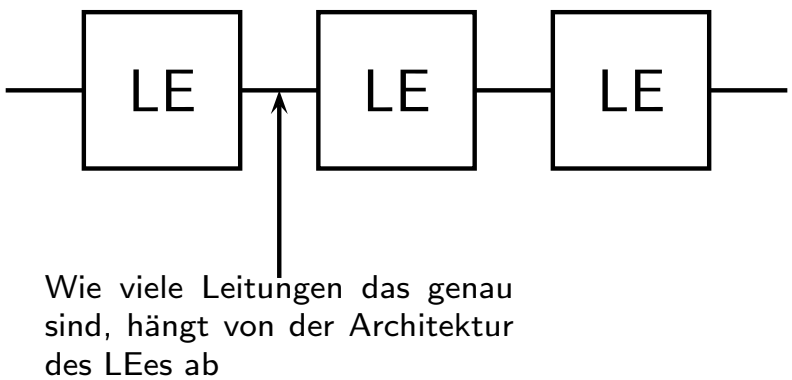
Zur Abbildung auf Ketten muss noch die Ketteneigenschaft erfüllt sein, d.h. ein LE hat **einen** Ausgang und **einen** Ausgang



ChainMap

Duplikation

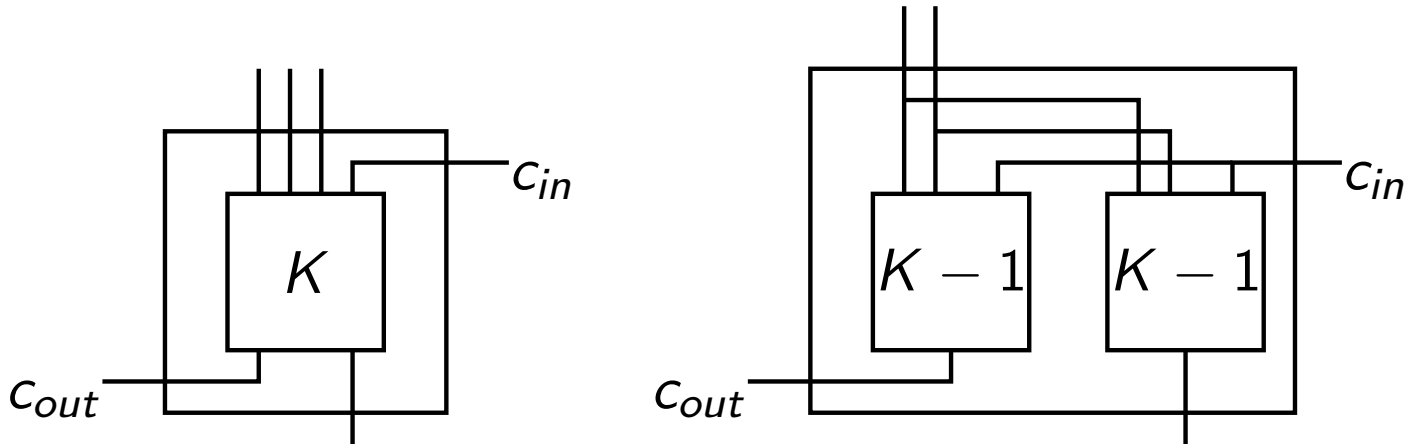
Zur Abbildung auf Ketten muss noch die Ketteneigenschaft erfüllt sein, d.h. ein LE hat **einen** Ausgang und **einen** Ausgang



ChainMap

Duplikation

Ab hier werden nur die folgenden LE-Strukturen betrachet

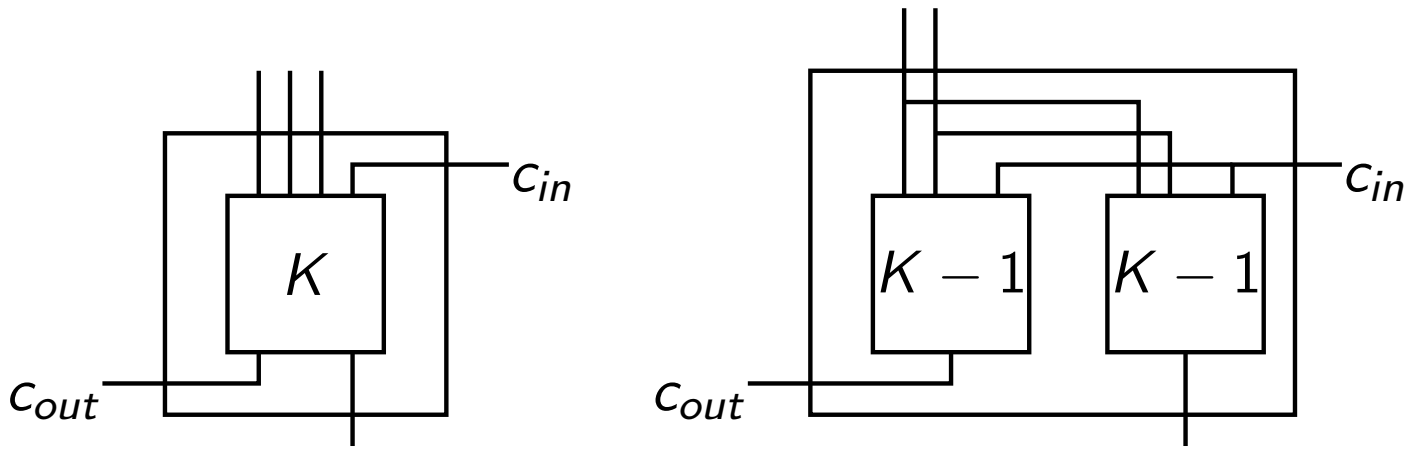


Der LUT-Graph aus dem Mapping darf also nur ein Binärgraph sein.

ChainMap

Duplikation

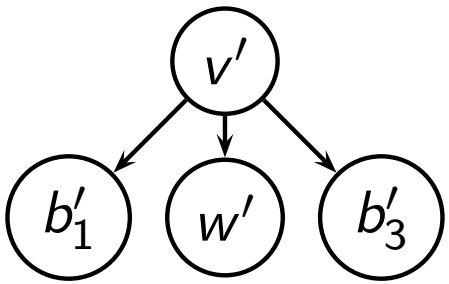
Ab hier werden nur die folgenden LE-Strukturen betrachet



Der LUT-Graph aus dem Mapping darf also nur ein Binärgraph sein. Frage: Was passiert, wenn er es nicht ist?

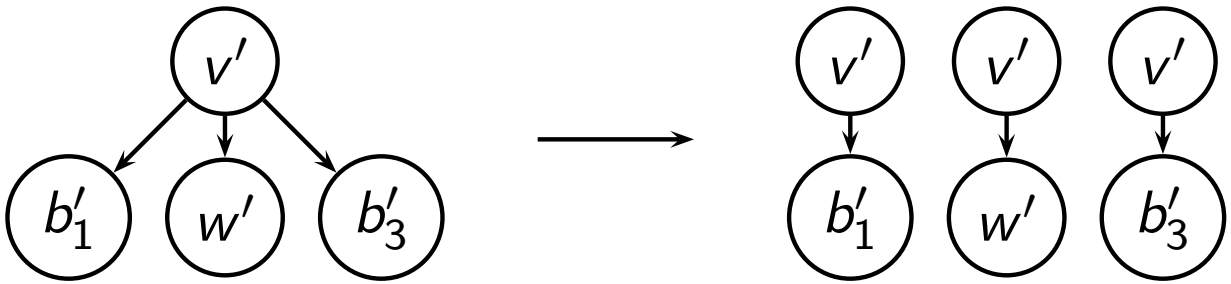
ChainMap

Duplikation, Beispiel



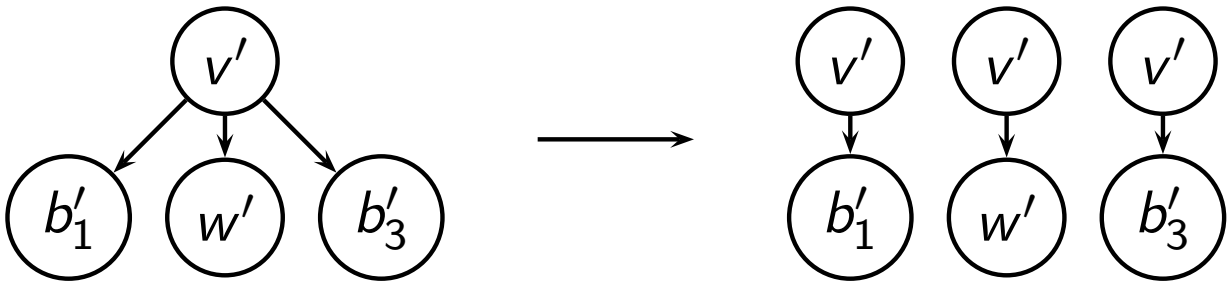
ChainMap

Duplikation, Beispiel



ChainMap

Duplikation, Beispiel



vollständige Duplikation ist **sehr** flächenaufwändig

ChainMap

Duplikation

Ein Knoten t und zwei Nachfolger u, v können in einer LUT beschrieben werden wenn

1. u und v nicht von einander abhängen

ChainMap

Duplikation

Ein Knoten t und zwei Nachfolger u, v können in einer LUT beschrieben werden wenn

1. u und v nicht von einander abhängen
2. wenn $\mathit{inputs}(u) = \mathit{inputs}(v), |\mathit{inputs}(u)| = K$, dann sind die Funktionen identisch

ChainMap

Duplikation

Ein Knoten t und zwei Nachfolger u, v können in einer LUT beschrieben werden wenn

1. u und v nicht von einander abhängen
2. wenn $\mathit{inputs}(u) = \mathit{inputs}(v), |\mathit{inputs}(u)| = K$, dann sind die Funktionen identisch
3. wenn $|\mathit{inputs}(u) \cup \mathit{inputs}(v)| < K$, so dürfen die Funktionen von u und v separat betrachtet werden

ChainMap

Duplikation

Ein Knoten t und zwei Nachfolger u, v können in einer LUT beschrieben werden wenn

1. u und v nicht von einander abhängen
2. wenn $\mathit{inputs}(u) = \mathit{inputs}(v), |\mathit{inputs}(u)| = K$, dann sind die Funktionen identisch
3. wenn $|\mathit{inputs}(u) \cup \mathit{inputs}(v)| < K$, so dürfen die Funktionen von u und v separat betrachtet werden
4. die Routing-Tiefe im (o.B.d.A) u-Zweig darf nicht größer werden

Ausblick

- Erweiterung des Konzeptes von ChainMap auf mehr Carry-Chain-Strukturen
- Automatisierung der Carry-Chain-Nutzung

Vielen Dank für Ihre Aufmerksamkeit