Heuristiken für das TechnologyMapping

Alexander Zorn

Geboren am 26. Mai 1996 in Bonn

2. April 2018

Bachelorarbeit Mathematik

Betreuer: Prof. Dr. Stephan Held

Zweitgutachter: YYYY YYYY

Forschungsinstitut für Diskrete Mathematik

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Inhaltsverzeichnis

TODO:
ZEILENUMBRÜCHE SCHÖNER WIRKEN LASSEN
EINLEITUNG AUF EINE SEITE STRECKEN UND SCHÖNER MACHEN!!
ICH ARBEITE VIEL MIT INVERTERN SOLLEN DIESE ALS EIGENSTÄNDIGE GATES DEFINIERT WERDEN ODER WEITERHIN ALS INPUTINVERTIERUNGEN? GENAUSO MIT DEN OUTPUTINVERTIERUNGEN
BILDER EINFÜGEN

1 Einleitung

Das Chipdesign ist ein Forschungsgebiet, welches in den letzten Jahrzehnten eine immer bedeutendere Rolle eingenommen hat. Es ist ein zu einem Projekt imenser Wichtigkeit und Beteiligung verschiedenster wissenschaftlicher Zweige (Mathematik, Physik, Informatik, Chemie etc.) geworden. Professor Korte/Vygen sagte einmal HIER ZITAT EINFUEGEN.
Die schwierige Aufgabe hierbei besteht darin einen booleschen Schaltplan von atemberaubender Größe auf einem wenige Quadratzentimeter großen Chip unterzubringen.
Ein Schaltplan (später als Circuit definiert) beschreibt hierbei eine implementierung einer Booleschen Funktion mithilfe kleiner Bauteile (später Gates). Eine solche lässt sich mit mehreren unterschiedlichen Bauplänen (Kandidaten) von Gates realisieren wobei jede Realisation Eigenschaften an Größe und Schnelligkeit (Delay) besitzt.
Die Aufgabe des TehnologyMapping ist es nun den Bauplan zu finden, welcher eine Kostenfunktion (bestehen aus Größe und Delay) optimiert.
In der Vorliegenden Arbeit wird ein PTAS (polynomial time approximation algorithm) für kleine Circuits vorgestellt und aus diesem eine Heuristik für die Anwendung auf dem gesamten Netz des Chips entwickelt.

WAS NOCH FEHLT : KLEINES BEISPIEL DER IMPLEMENTIERUNG GEBEN TERMINOLOGIE VERBESSERN AUF BENUTZTE ARBEITEN VERWEISEN

2 Terminologie & grundlegender Algorithmus

Es folgen ein paar grundlegende Definitionen zur Beschreibung des Problems.

Definition 2.1. Boolesche Variable und Funktion:
Eine boolesche Variable ist eine Variable mit Werten in $\{0,1\}$.
Sei $n, m \in \mathbb{N}$. Eine boolesche Funktion ist eine Funktion $f : \{0,1\}^n \to \{0,1\}^m$ mit n inputs und m outputs.

Definition 2.2. Gate und Library:
Ein Gate g mit Eingangsgrad $n \in \mathbb{N}$ ist ein Tripel $(f_g, d_g, area_g)$. Hierbei

sind $d_g, area_g \in \mathbb{R}_{\geq 0}$. Des Weiteren gilt f_g ist eine boolesche Funktion mit $f_g : \{0,1\}^n \to \{0,1\}$.
Eine Library L ist eine Menge von Gates und sei $fanin_{max} := max\{arity(g)|g \in L\}$.

$area_g$ gibt die Größe des physikalischen Bauteils an und d_g beschreibt die Zeit die ein Signal braucht um von den inputs des Gates zu seinem Output zu gelangen. Dieser Wert lässt sich noch weiter differenzieren indem man $d_g \in \mathbb{R}^n$ wählt und somit Zeiten für jeden der Inputs angeben werden kann.

Definition 2.3. Circuit:
Ein Circuit ist ein gerichteter kreisfeier Graph (directed acyclic graph DAG) mit folgenden Eigenschaften. Jeder Knoten gehört zu einer der aufgelisteten Kategorien:

- **Input** Knoten mit Eingangsgrad Null.
- **Gates** mit mindestens einer eingehenden Kante und ausgehenden Kante. Diese korrespondieren zu der Definition oben mit dem Zusatz dass an jedem der Inputs optional ein Inverter liegen kann.
- **Outputs** mit genau einer eingehenden Kante und keiner ausgehenden.

Ein Gate mit mehr als einer ausgehenden Kante wird auch Highfanoutgate genannt.
Ein Circuit realisiert durch Verschachtelung der booleschen Funktionen seiner Gates ebenfalls eine boolesche Funktion.
Zwei Circuits heißen äquivalent, wenn sie die gleiche boolesche Funktion realisieren.

In einem Circuit lassen sich Teilgraphen durch ein Gate der Library austauschen. Voraussetzung für einen solchen Tausch ist, dass der veränderte Circuit äquivalent zu dem originalen ist. Dies sicher die folgenden Definitionen.

Definition 2.4. Match und Kandidat:
Sei g ein Gate in einem Circuit C. Ein (invertiertes) Match m ist ein Tupel (p_m, I_m, f_m, inv_m) welches folgendes enthält:

- Ein Gate p der Library
- Eine Menge X von Knoten aus der Circuit und eine Bijektion $f : X \to inputs(p)$
- Ein Funktion $inv : inputs(p) \to \{not\_inv, inv\}$

So dass der Circuit C', welcher durch den Austausch des Sub-Circuits von X bis g durch das Match (mit den durch inv definierten Invertern an den

Inputs) entsteht, äquivalent zu C ist. Ein invertiertes Match auf g ist ein Match auf g mit einem Inverter an jedem seiner Outputs.
Ein (invertierter) Kandidat auf g besteht aus einem (invertierten) Match auf g und einem Kandidaten für jeden Input Knoten von g (welcher kein Input von C ist).

Definition 2.5. Circuit-Kandidat: Sei C ein Circuit mit Outputknoten Menge O. Eine Circuit-Kandidat K von C ist eine Menge von Kandidaten, sodass $\forall o \in O \, \exists! h \in K : h$ ist Kandidat von o und an jedem Knoten von C an dem sich mehrere Kandidaten überschneiden ist dasselbe Match gewählt.

Folgendes Beispiel visualisiert die vorherigen Definitionen.
BILD EINSETZEN

Ein Circuit-Kandidat C ist eine Möglichkeit den Circuit physikalisch zu realisieren. Wie bereits in der Einleitung bemerkt gilt es nun den besten Kandidaten auf C auszuwählen. Dafür ist ein Maß für Implementierungen von Circuits notwendig. Es folgen zwei geläufige Beispiele. In der Praxis (und im späteren Verlauf dieser Arbeit) wird in der Regel eine convex-Kombination aus beiden verwendet.

Definition 2.6. Area und Delay eines Kandidaten:
Sei C ein Circuit und K ein Circuit-Kandidat auf C. Dann gilt:

- $area(K) = \sum_{g \in gates(C)} (a_g + \sum_{i \in inputs(g)} \mathbb{1}_{\{inv_g(i) == inv\}} area_{inv})$
  wobei $area_{inv}$ die Größe eines Inverters ist.
- $AT(K) =$
  $\max_{k \in can(K)} \{ \max_{i \in inputs(k)} \{ d_{gate(k)} + \mathbb{1}_{\{inv_g(i) == inv\}} d_i + AT(inp\_can(k,i)) + d_{w(k,i)} \} \}$

Wobei $can(K)$ die Menge der Kandidaten von K sind und $inputs(k)$ sind die Inputknoten des Outputknoten des Kandidaten k. Des Weiteren ist d_i das Delay eines Inverters und $d_{w(k,i)}$ das Delay der Kante zwischen den Knoten k und i. $inp\_can(k,i)$ gibt den Kandidaten des i'ten Inputs von k zurück.

Das Delay (AT) gibt an wann das letzte Signal aus einem der Outputs des Circuit kommt.

AB HIER LUCAS VORLAGE

Definition 2.7. Library: Eine Library ist eine Menge L von Gates (boolesche Funktionen) mit zwei Abbildungen $d, \text{area} : L \to \mathcal{R}_{\geq 0}$, die jedem Gate sowohl eine Verzögerung d_l, als auch eine Fläche area_l zuordnen.

Definition 2.8. Circuit: Ein Circuit C auf der Library L ist ein zusammenhängender gerichteter azyklischer Graph (DAG), bei dem jeder Knoten einer dieser 3 Arten entspricht:

- einem Inputknoten ohne eingehende Kanten
- einem Gate aus L mit ≥ 1 eingehenden und ≥ 1 ausgehenden Kanten
- einem Outputknoten ohne ausgehende Kanten

Jeder Gateknoten kann an jeder seiner eingehenden Kanten einen Inverter vorschalten. Für einen Knoten v sei fanin(v) die Zahl seiner eingehenden, fanout(v) die Zahl seiner ausgehenden Kanten. Knoten mit fanout(v) > 1 heißen Highfanoutknoten. Wir betrachten vorerst nur Circuits mit exakt einem Outputknoten.

Definition 2.9. cone: Für einen Knoten g aus einem DAG S bezeichne

$$cone\,(g) := S\,[V \cup \{g\}]\,, V = \{v \in V(S) : \exists v\text{-}g\text{-Weg in } S\}$$

Sowie für eine Knotenmenge G sei $cone(G) := S\,[\cup_{g \in G} V(cone(g))]$

Algorithmus : TechnologyMapping auf einer Arboreszens

Input : Circuit C kreisfrei mit finalem Output o, Library L verfügbarer Gates

```
1 bester_kandidat[] ← ∅
2 bester_inv_kandidat[]← ∅
3 foreach Knoten n ∈ V(G) in topologischer Ordnung do
4 |   berechne alle (invertierte) Matches auf n
5 |   foreach Match m auf n do
6 |   |   Berechne besten Kandidaten mit m auf n
7 |   |   Update best_(inv)_kandidaten
8 best_final ← bester_kandidat[o]
9 Implementiere C entsprechend best_final
```