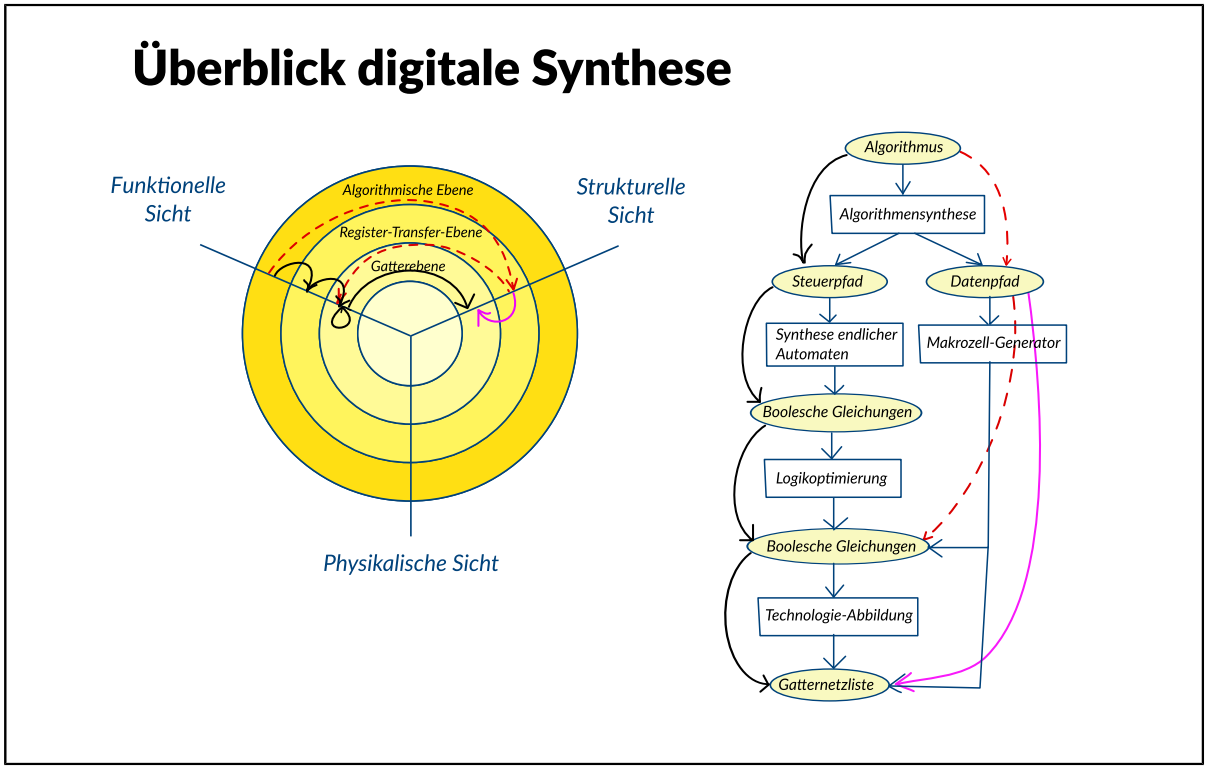
Electronic Design Automation (EDA)

Technology Mapping

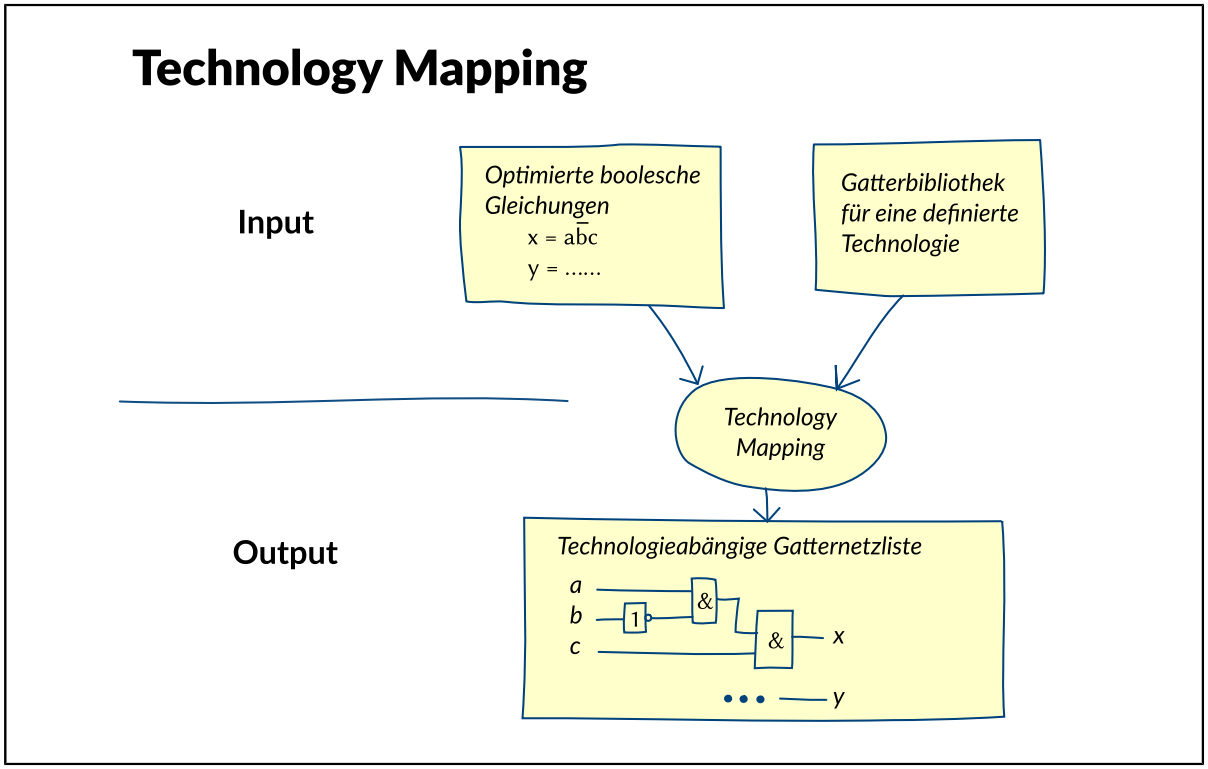
Überblick digitale Synthese
Technology Mapping
Abbildung durch die Abdeckung eines Baumes
Partitionierung des DAG
Dekomposition und Abdeckung
Beispiel

Date of generation: 26.10.2026

Technology Mapping: Überblick digitale Synthese



Technology Mapping: Technology Mapping



Die Technologie-Abbildung bildet den Abschluss der Synthese. Da Flipflops und Treiber in der Regel unmittelbar durch entsprechende Elemente einer Bibliothek ersetzt werden, ist die Hauptaufgabe der Technologie-Abbildung die Optimierung kombinatorischer Schaltungsteile. Ausgangspunkt hierfür sind zwei- oder mehrstufige logische Gleichungen. In diesem Kapitel werden nur Methoden für mehrstufige Darstellungen beschrieben. Zweistufige Darstellungen dienen im Allgemeinen als Eingabe für PLA-Generatoren und werden hier nicht behandelt.

Die booleschen Gleichungen werden mit verfügbaren Zellen einer Bibliothek implementiert, die zu einer bestimmten Technologie gehört. Die Bibliotheken enthalten Gatter (NOR, Inverter,...) und komplexere Elemente wie Komparatoren, Halbaddierer o.ä.. Die Gatter sind als boolesche Funktionen beschrieben und durch ihre Fläche und ihre Verzögerungszeiten charakterisiert. Die Verzögerungszeit setzt sich meist aus einem lastabhängigen und einem lastunabhängigen Anteil zusammen.

Technology Mapping: Abbildung durch die Abdeckung eines Baumes

Abbildung durch Abdeckung eines Baumes

- Primäres Ziel ist Kosten(Flächen)-Optimierung.
- Zur Schaltungsmodellierung wird in der Regel ein gerichteter azyklischer Graph benutzt (Directed Acyclic Graph = DAG).
- Ziel ist es, den DAG der Schaltung durch Subgraphen abzudecken, die die in der Bibliothek implementierten Basisfunktionen darstellen.
- Eine exakte Lösung des Problems ist zu aufwendig, daher Lösung durch regelbasierte oder andere heuristische Verfahren.
- Regelbasierte Verfahren finden optimale Lösungen in überschaubaren Teil-DAGs.
- Dies führt zu einem allgemeinen Baum-Abdeckungsproblem:
  1. Aufbau des DAG der Schaltung
  2. Partitionierung in Bäume
  3. Dekomposition der Bäume in Basisfunktionen
  4. Abdeckung aller Bäume der Schaltung mit Zellen aus der Bibliothek

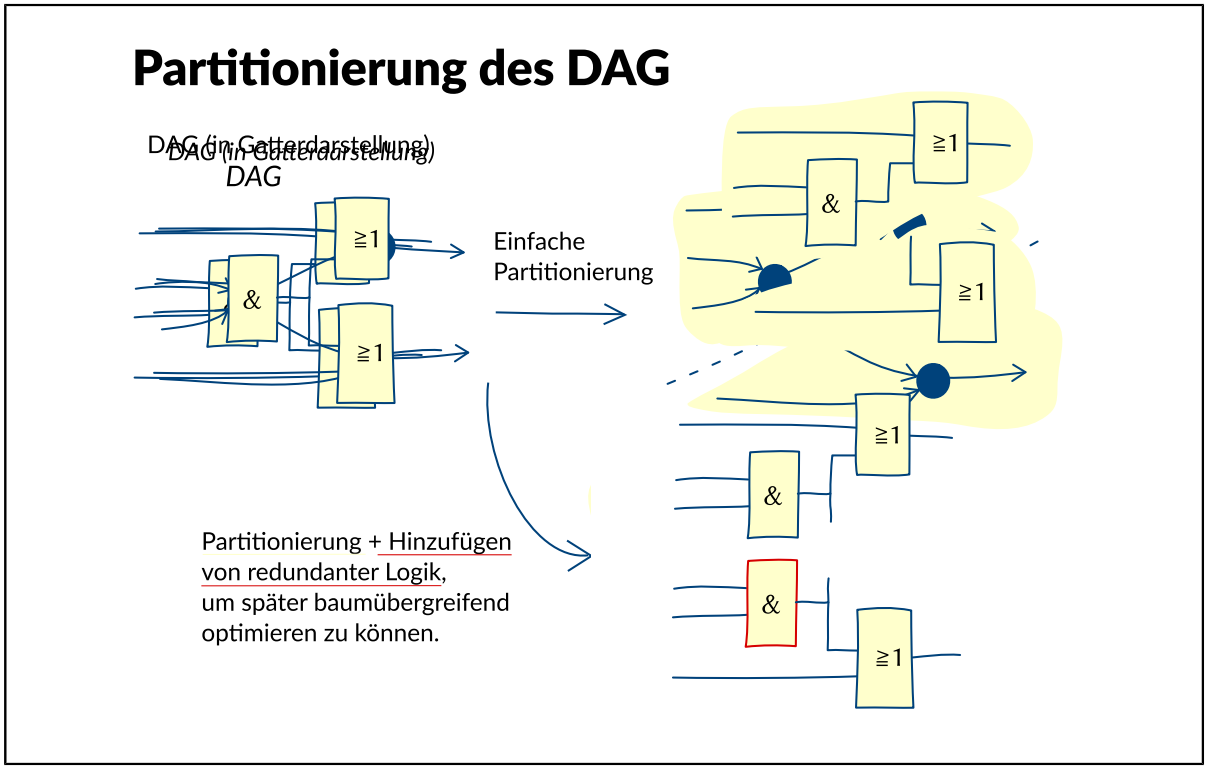
In erster Linie wird eine Implementierung gesucht, deren Kosten möglichst gering sind, indem eine flächenminimale Zellenkombination gesucht wird. Bei zeitkritischen Pfaden können stattdessen auch die schnellsten verfügbaren Zellen ausgewählt werden. Die Berücksichtigung der Performance in Form von Verzögerungszeiten während der Abbildung ist schwierig, da der lastabhängige Teil der Verzögerung erst am Ende der Abbildung bestimmt werden kann.

Häufig verwendete **heuristische** Methoden basieren auf einer Baum-Abdeckung. Diese besteht aus vier Schritten:

- Aufbau des DAG
- Partitionierung des Graphs (der Schaltung) in Bäume
- Dekomposition der Bäume (Schaltungsteile) in Basisfunktionen
- Abdeckung aller Bäume (der Schaltung) mit Zellen aus der Bibliothek

Die Abdeckung eines DAGs ist ein NP-hartes Problem, seine exakte Lösung ist daher zu aufwendig. Dagegen sind Abdeckungsalgorithmen für Bäume praktikabel. Daher wird der DAG in einen Wald von Bäumen zerlegt.

Technology Mapping: Partitionierung des DAG



Ein einfacher Weg, den DAG in Bäume zu zerlegen, besteht darin, alle Verzweigungen aufzutrennen und anschließend alle Bäume einzeln abzudecken. Der Nachteil davon ist, dass beim Abdecken nur lokal innerhalb eines Baums optimiert werden kann und nicht baumübergreifend.

Eine bessere Partitionierung kann erreicht werden, indem an den aufgetrennten Stellen zusätzliche Logik eingefügt wird. Der Pfad von den Eingangsvariablen bis zum aufgetrennten Knoten wird für jede Verzweigung hinzugefügt. Das erzeugt zwar Redundanz, ermöglicht aber eine baumübergreifende Optimierung.

Technology Mapping: Dekomposition und Abdeckung

Dekomposition und Abdeckung

Dekomposition des DAG der Schaltung in eine kanonische Darstellung mit Basisfunktion z.B. NAND, NOR mit zwei Eingängen (Subject Graph).

Bibliothek enthält alle Basisfunktionen als sogenannte Pattern Graphs
-> triviale Abdeckung existiert.

Ziel ist, durch Verwendung komplexer Zellen der Bibliothek (Komplexgatter) eine flächengünstige Lösung zu finden.

Die **Abdeckung** erfolgt in drei Schritten:

- **Matching:** Erstellung einer Liste aller möglichen Matchings für alle Teilbäume (Ersatz von Teilen des Subject Graphs durch isomorphe Pattern Graphs)
- **Optimierung:** Bestimmung des Matchings mit den niedrigsten Kosten für jeden Teilbaum
- **Akkumulation (Mapping):** Top-Down-Abdeckung des Gesamtbaumes ausgehend von der optimalen Zelle für die Wurzel des Baumes

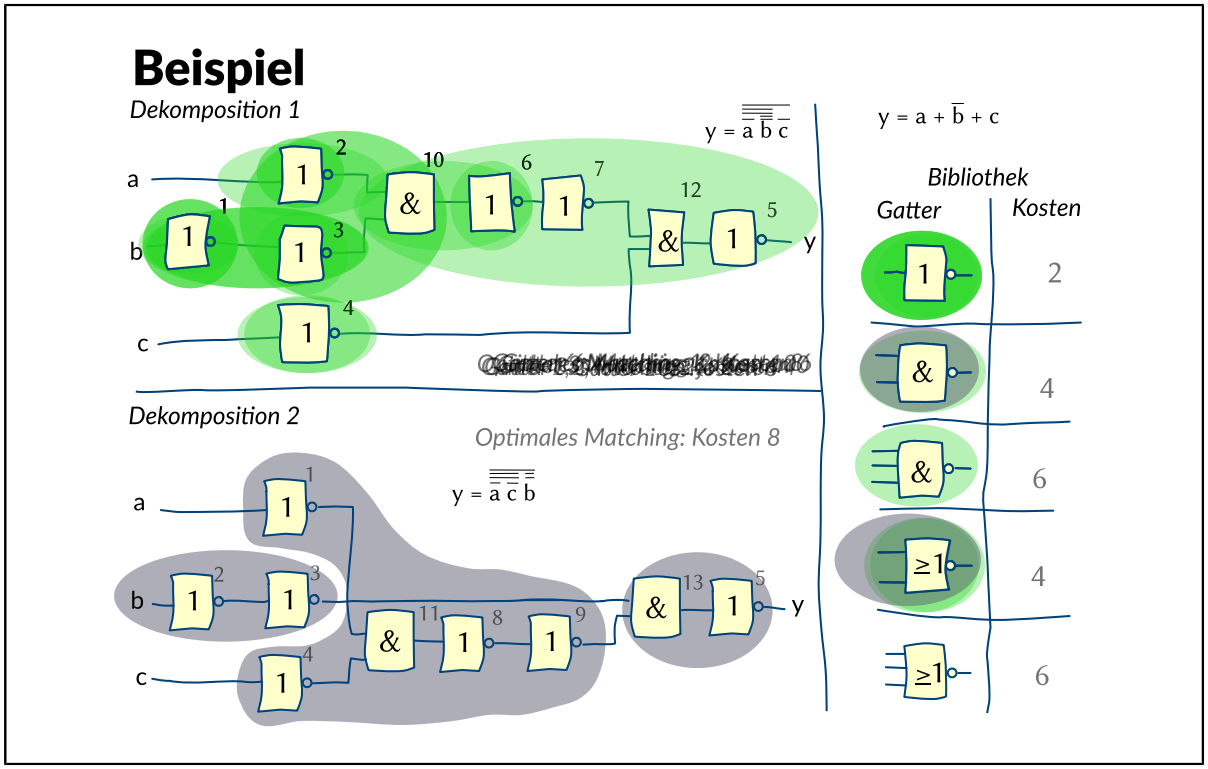
Für die eigentliche Abdeckung erfolgt eine Dekomposition der Schaltung in eine kanonische Darstellung mit Basisfunktionen (z.B. NAND-, NOR-Gatter mit 2 Eingängen und Invertern) für den DAG (Subject Graph genannt) und die Bibliothek (Pattern Graph genannt). Da die Bibliothek in der Regel auch diese Basisfunktionen enthält, gibt es mindestens eine triviale Abdeckung. Ziel ist es jedoch durch Verwendung komplexer Zellen der Bibliothek (Komplexgatter), z.B. Gatter mit mehr als zwei Eingängen, eine flächengünstigere Lösung zu finden.

Der Lösungsraum für die Abdeckung wird durch die Partitionierung und die Dekomposition eingeschränkt. Die Lösung erfolgt in zwei Schritten mit einem dynamischen Algorithmus:

Matching mit **Optimierung** Für alle Teilbäume eines Baums wird eine Liste mit allen möglichen Matchings erstellt. D.h. Teilbäume des Subject Graphs werden durch isomorphe Pattern Graphs ersetzt. Dies ist ein rekursives Bottom-Up-Verfahren, das das optimale Matching für jeden Teilbaum findet. Dabei wird von den Eingängen des Subject Graphs schrittweise zu den Ausgängen vorgegangen. Für jede zugeordnete Zelle eines Knotens des Baums werden die Kosten der Zelle und der optimalen Matchings der Eingänge dieser Zelle berechnet. Am Ende wird das optimale Matching an der Wurzel des Baums bestimmt (Matching mit den niedrigsten Kosten).

Akkumulation Hier werden in einem Top-Down-Verfahren die Ergebnisse der Optimierung genutzt, um eine Abdeckung des gesamten Baums zu erzielen. Ausgehend von der optimalen Zelle der Wurzel des Baumes, werden rekursiv die Teilbäume der Eingänge der zugeordneten Zelle zugeordnet.

Technology Mapping: Beispiel



Das Bild zeigt zwei Dekompositionen einer Schaltung (eines Baumes) in AND-Gatter mit zwei Eingängen und Inverter. Als einfaches Beispiel für eine Abdeckung wird die obere Dekomposition betrachtet. Die Bibliothek beinhaltet NAND- und NOR-Zellen mit bis zu drei Eingängen und Inverter. Die Kosten werden vereinfacht durch die Anzahl der Transistoren der Zelle bestimmt und setzen sich für die einzelnen Matchings zusammen aus den Kosten der zugeordneten Zelle und den Kosten für die optimalen Matchings der Pfade bis zu den Eingängen dieser Zelle.

1. Matching

Zuerst werden die Gatter an den Eingängen betrachtet. Für die Gatter 1, 2 und 4 gibt es jeweils nur ein Matching, nämlich einen Inverter, der hier die Kosten zwei hat.

Für Gatter 3 gibt es zwei Matchings: Zum einen Weglassen der beiden Inverter 1 und 3 mit den Kosten null, oder einen Inverter. Die Kosten für das zweite Matching setzen sich zusammen aus der zugeordneten Zelle, einem Inverter mit den Kosten zwei, und dem Pfad zum Eingang dieser Zelle. In dem Beispiel besteht dieser Pfad aus dem Gatter 1, für das gerade die Kosten zwei ermittelt wurden. Die Kosten insgesamt betragen also vier.

Für Gatter 10 gibt es nur ein Matching und zwar eine 2-fach NOR-Zelle, die Gatter 2, 3 und 10 abdeckt (not(x1+x2)=(not x1)(not x2)). Die Kosten davon sind vier für die NOR-Zelle und wiederum zwei für den Eingang von Gatter 3, d.h. Gatter 1. Insgesamt hat dieses Matching die Kosten sechs.

Für Gatter 6 gibt es zwei Matchings: Das eine ist ein Inverter mit den Kosten acht (zwei für den Inverter und sechs für dessen Eingang, das Matching für Gatter 10) und das andere ein 2-fach NAND, das die Gatter 10 und 6 abdeckt. Letzteres hat insgesamt die Kosten sechs (vier für die NAND-Gatter plus zwei für Gatter 2 plus Null für die sich aufhebenden Gatter 1 und 3). Da das erste teurer ist, wird es verworfen. So werden alle weiteren Gatter bis zum Ausgang bearbeitet.

2. Akkumulation

Danach werden ausgehend vom optimalen Matching des letzten Gatters die optimalen Matchings für dessen Eingänge bestimmt, und danach die Matchings für dessen Eingänge usw.. Für das Beispiel

wird das Ausgangsgatter am besten durch ein 3-fach NAND abgedeckt, das die Gatter 5, 6, 7, 10 und 12 beinhaltet. Diese Zelle wird als Teillösung gewählt. Danach müssen die optimalen Matchings für die Eingänge dieser Zelle der Lösung hinzugefügt werden. In diesem Falle sind dies die optimalen Matchings für die Gatter 2, 3 und 4 und das sind zwei Inverter für Gatter 2 und 4. Die Kosten der Lösung sind insgesamt zehn.

Dieser Algorithmus liefert für die zweite Dekomposition eine andere Lösung, nämlich ein 2-fach NOR, das die Gatter 1, 2, 3, 4, 8, 9 und 11 abdeckt, und ein 2-fach NAND für den restlichen Teil der Schaltung. Diese Lösung hat die Kosten acht und ist damit billiger als die erste Lösung.

Nachteile dieses Verfahrens sind, dass das Ergebnis von der Dekomposition abhängt und dass einige Gatter sich nicht als Baum darstellen lassen (wie z.B. XOR). Es gibt einige Erweiterungen dieses Verfahrens, die diese Nachteile vermeiden, dafür aber aufwendiger sind oder den Lösungsraum weiter einschränken.