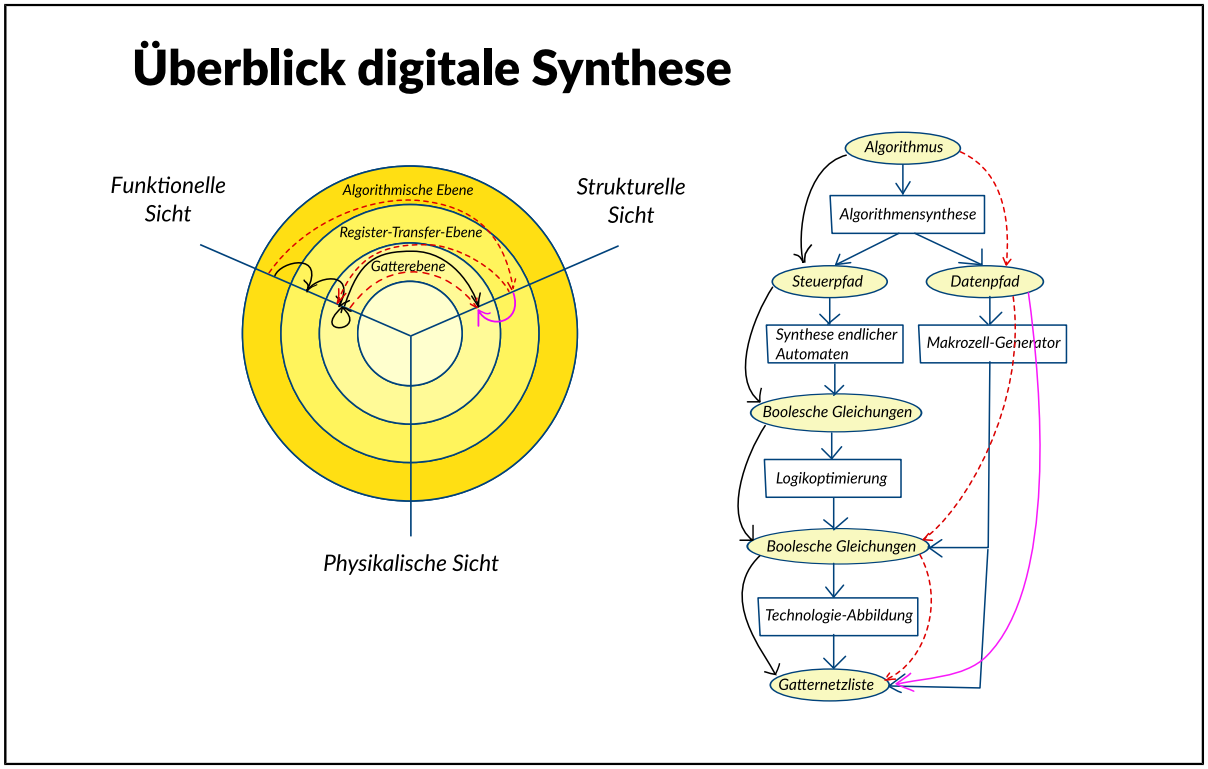
Electronic Design Automation (EDA)

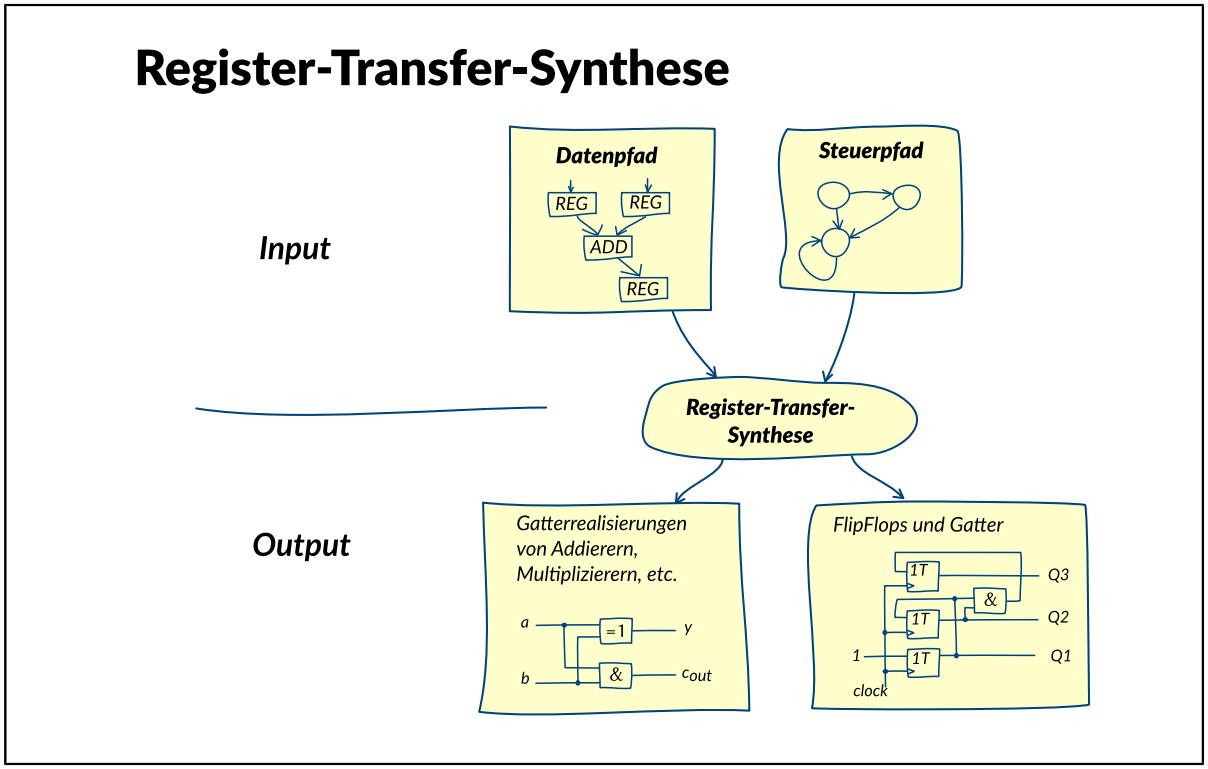
Register-Transfer-Synthese

Date of generation: 30.5.2023

Register-Transfer-Synthese: Überblick digitale Synthese



Register-Transfer-Synthese: Register-Transfer-Synthese



Die Register-Transfer-Synthese setzt eine Schaltung auf Register-Transfer-Ebene in eine funktional äquivalente Schaltung auf Gatterebene um. Das Zeitverhalten der Register-Transfer-Ebene wird taktgenau beibehalten. Zur Synthese der Elemente des Datenpfades werden parametrisch steuerbare Generatoren (Makrozell-Generatoren*) eingesetzt, z.B. bei der Abbildung von Operationen wie Multiplikation oder Addition. Zur Synthese des Steuerpfades wird auf spezielle Synthese-Algorithmen für die Abbildung endlicher Automaten zurückgegriffen.

*: Bei der Generierung von Makrozellen handelt es sich eigentlich um einen Syntheseschritt. In der Literatur wird jedoch einheitlich der Begriff Makrozellen- oder Modul-Generator verwendet. Daher behalten wir diesen Begriff entgegen der sonst von uns verwendeten Nomenklatur bei.

Register-Transfer-Synthese: Makrozellgenerator

Makrozellgeneratoren

- Makrozellen:
  Addierer, Multiplizierer, Speicher, Schieberegister, Zähler, ...

- Höhere Komplexität als Standardzellen

- Höhere Regularität/wenige Parameter (z.B. Bitbreite, Wortlänge)

- Unterschiedliche Realisierungen mit unterschiedlichen Eigenschaften:

  Optimierung von:
  - Performance (Verzögerungszeit)
  - Kosten (Fläche),
  - Leistungsaufnahme,
  - ...

Zu den wesentlichen Elementen eines Datenpfades gehören Teilschaltungen wie Addierer, Multiplizierer, Speicher sowie weitere regulär aufgebaute Logikblöcke wie Schieberegister oder Zähler. Diese Teilschaltungen bezeichnen wir als Makrozellen. Makrozellen haben im Gegensatz zu Basis- bzw. Standardzellen eine erheblich höhere Komplexität.

Makrozellen zeichnen sich in der Regel durch eine hohe Regularität aus. Daher lassen sie sich durch wenige Parameter wie die Bitbreite bei einem Addierer oder die Wortlänge und die Anzahl der Worte bei einem Speicher beschreiben. Diese Parameter reichen aus, um mit Hilfe eines Makrozell-Generators aus einfachen Grundzellen die gewünschte Teilschaltung zu erzeugen. Beispielsweise können Speicher aus 1-Bit-Grundzellen oder Addierer aus 1-Bit-Voll-Addierern zusammengesetzt werden.

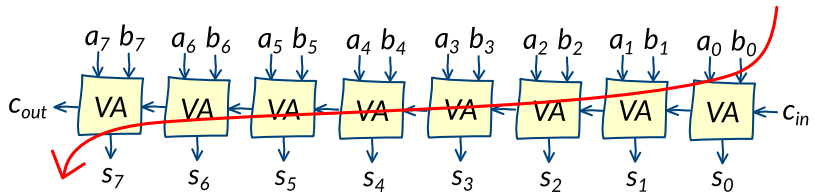
Beispiel Addierer: Variante 1

8-bit-Ripple-Carry-Addierer

Kritischer Pfad durch:
- 8 Volladdierer

Flächenbedarf:
- 8 Volladdierer

	klein	schnell
8 bit	☒	☐
16 bit	☐	☐



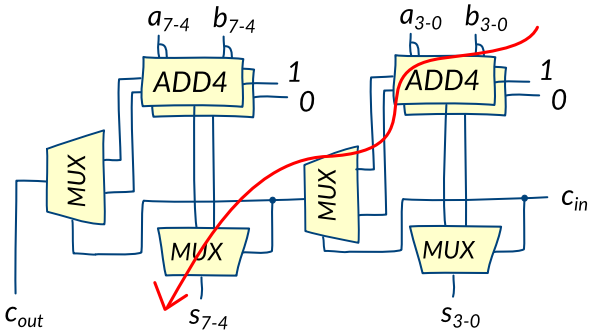
Beispiel Addierer: Variante 2

8-bit-Carry-Select-Addierer

Kritischer Pfad durch:
- 4 Volladdierer (ADD4)
- 1 Carry-Select-Logik (CS)
- 1 Multiplexer (MUX)

Flächenbedarf:
- 16 Volladdierer
- 2 Multiplexer
- 2 CS

	klein	*schnell*
8 bit	☐	☒
16 bit	☐	☐



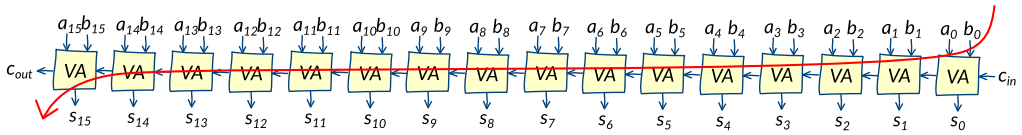
Beispiel Addierer: Variante 3

16-bit-Ripple-Carry-Addierer

Kritischer Pfad durch:
- 16 Volladdierer

Flächenbedarf:
- 16 Volladdierer

	klein	schnell
8 bit	□	□
16 bit	☒	□



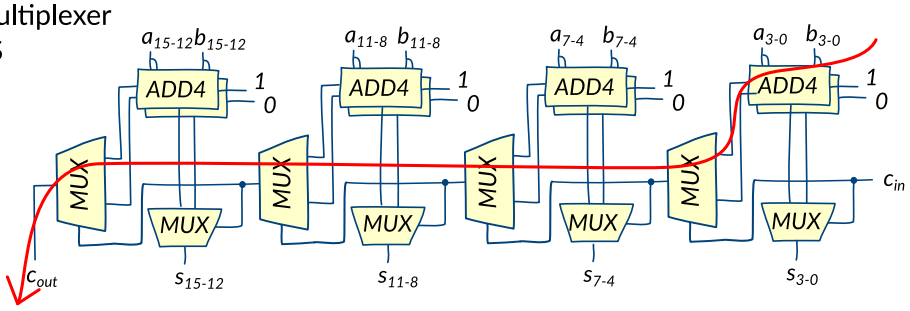
Beispiel Addierer: Variante 4

16-bit-Carry-Select-Addierer

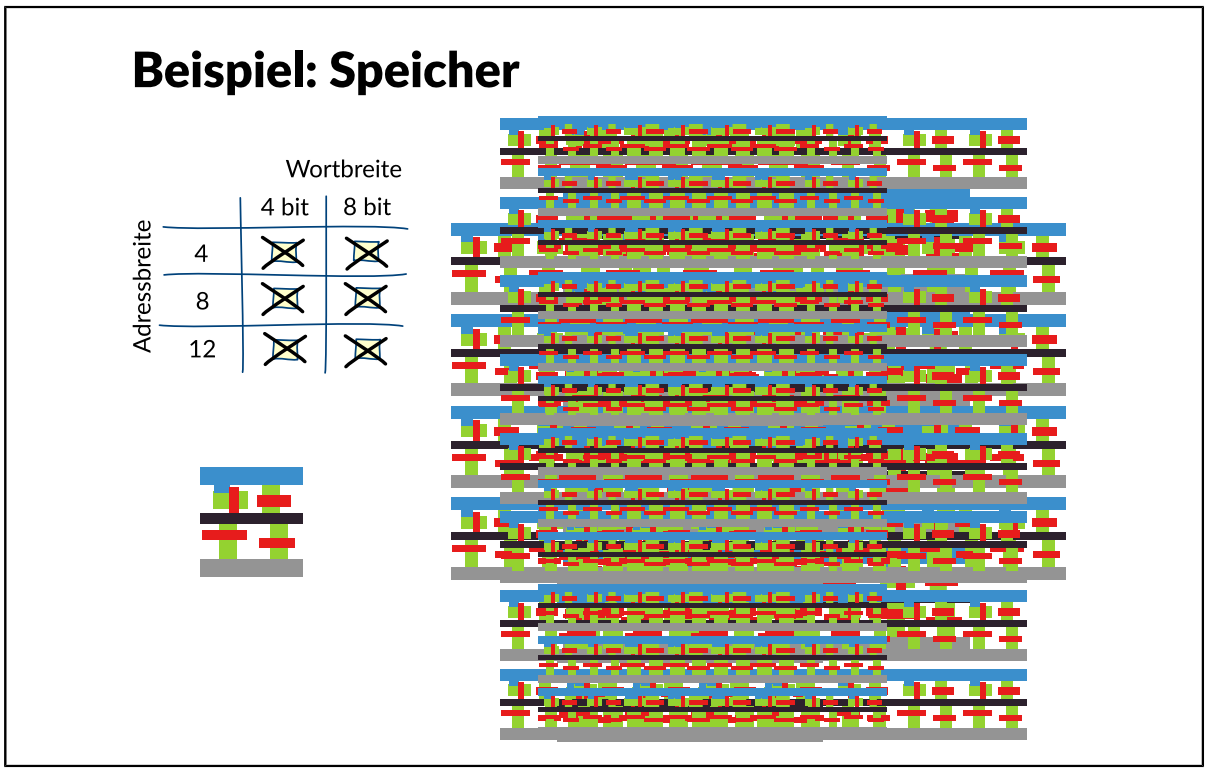
Kritischer Pfad durch:
- 4 Volladdierer (ADD4)
- 3 Carry-Select-Logik (CS)
- 1 Multiplexer (MUX)

Flächenbedarf:
- 32 Volladdierer
- 4 Multiplexer
- 4 CS

	klein	schnell
8 bit	☐	☐
16 bit	☐	☒



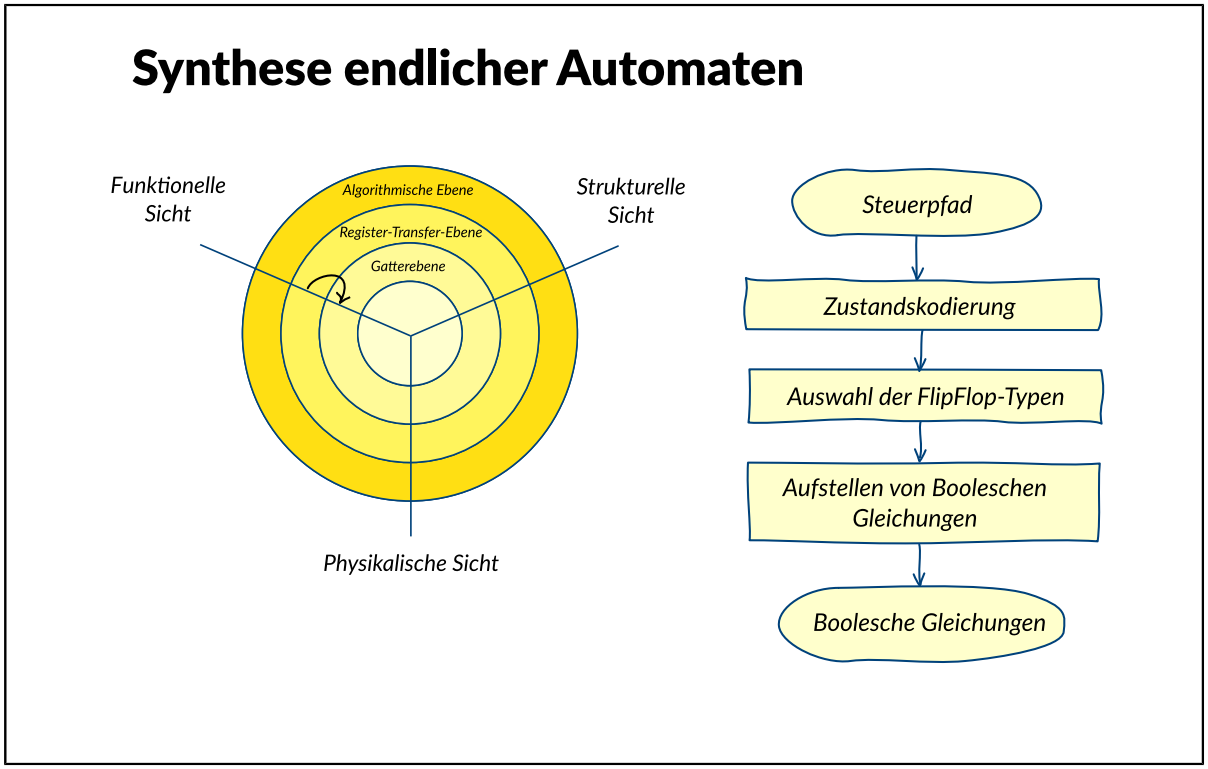
Register-Transfer-Synthese: Beispiel Speicher



Bei Strukturen wie Speichern (DRAM, SRAM, Flash) sind die Grundzellen stark spezialisiert. Das Layout wird manuell entworfen und ist noch optimiert. Ein Makrozell-Generator liefert dann nicht nur eine (zumeist triviale) Darstellung in funktionaler oder struktureller Sicht auf Gatterebene, sondern auch bereits ein Layout. Übliche Darstellungen sind: Ein zeitlich exaktes Modell auf Gatterebene für die Schaltungsverifikation, ein so genanntes Abstract, das im Wesentlichen die äußere Form und die Lage der Anschlussfelder beschreibt und das als Platzhalter für das Makrozell-Layout bei der Platzierung und Verdrahtung der Gesamtschaltung verwendet wird, und schließlich das Layout selbst, das nach Abschluss aller Entwurfsschritte den Platz des Abstracts einnimmt.

Mit einem entsprechenden Makrozell-Generator kann ein Halbleiterhersteller auch bei sehr zeit- und flächenkritischen Makrozellen wie bei eingebetteten dynamischen Speicherfeldern eine korrekte Funktion gewährleisten, weil er sein spezielles Know-How in den Makrozell-Generator integrieren kann. Im Allgemeinen wird man bei Speichern eine Kombination finden: Optimierte Zellen für das Speicherfeld und Standardzellen für die Adress-Dekoder-Logik.

Register-Transfer-Synthese: Synthese endlicher Automaten

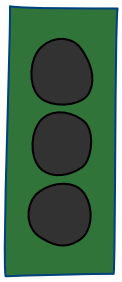


Aufgabe der Synthese endlicher Automaten ist es, Automaten in ein System von booleschen Gleichungen umzuwandeln. Die Anzahl der Zustände sei bereits minimiert und der Automat als Zustandsgraph beschrieben. In diesem Graphen sind die einzelnen Zustände als Knoten und die Zustandsübergänge, die auf Änderungen der Eingangsvariablen folgen, als gerichtete Kanten dargestellt.

Um zu booleschen Gleichungen zu gelangen, werden zunächst die Zustände kodiert, Flipfloptypen ausgewählt und das Übergangs- und das Ausgangsschaltnetz in booleschen Gleichungen beschrieben.

Register-Transfer-Synthese: Zustandskodierung

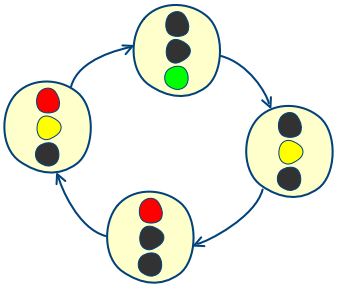
Zustandskodierung



Binär	*Gray*	*One-Hot*	*"Ampel"*
00	*00*	*0001*	*001 (grün)*
01	*01*	*0010*	*010 (gelb)*
10	*11*	*0100*	*100 (rot)*
11	*10*	*1000*	*110 (rotgelb)*
Minimale Anzahl von Zustandsbits	*Nur ein Bitwechsel bei Zustandswechsel*	*Pro Zustand ein FlipFlop*	*Zustände = Ausgänge*

Kleine Automaten:
- Finden der optimalen Kodierung möglich

Große Automaten:
- Heuristiken nötig



Mit der Zustandskodierung wird den Zuständen des Automaten eine binäre Repräsentation zugeordnet. Dadurch können die Zustände durch eine Menge von Flipflops realisiert werden. Ziel der Zustandskodierung ist es, minimale Kosten zu erzielen. Für PLA-Realisierungen soll die Anzahl der Produktterme nach der zweistufigen Logikminimierung minimal sein, für mehrstufige Realisierungen die Anzahl der Literale in der faktorisierten Form.

Für die Kodierung kleiner Automaten gibt es drei Standardverfahren:

- Binäre Kodierung
- Gray-Kodierung
- One-Hot-Kodierung

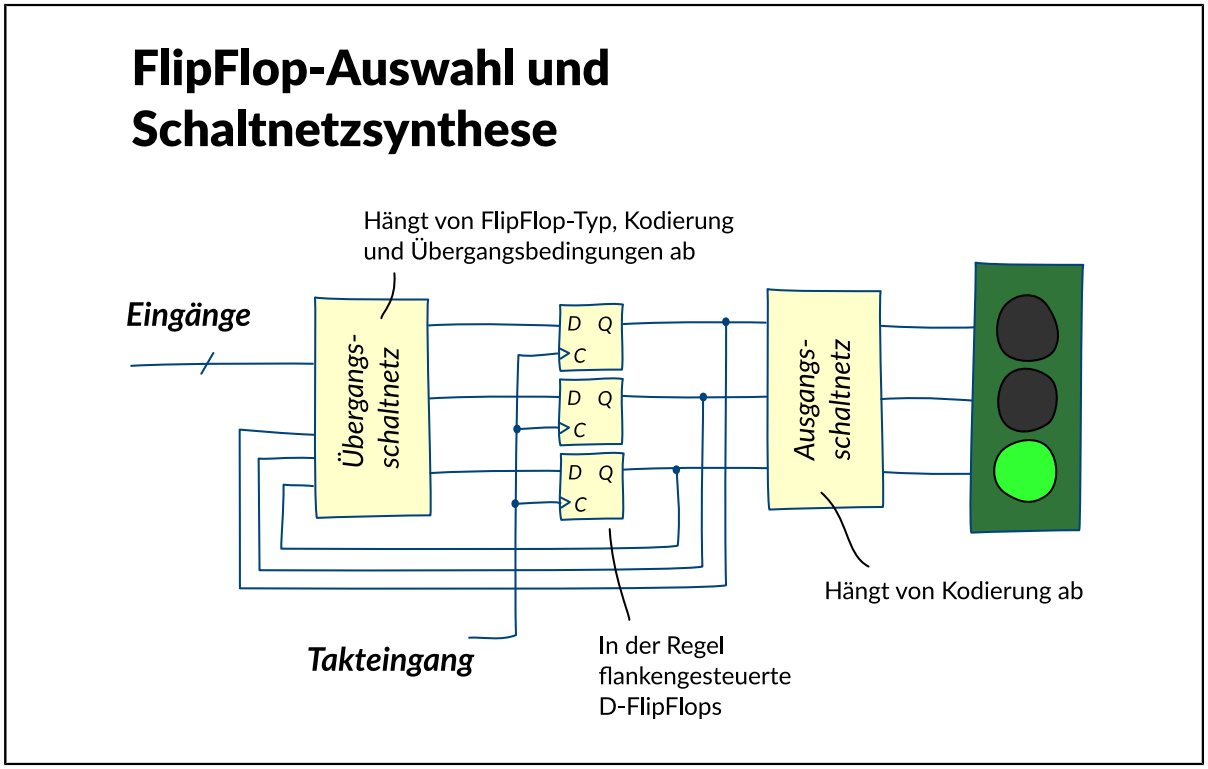
Bei der binären Kodierung werden die Zustände binär durchnummeriert. Mit n Flipflops lassen sich so 2^n Zustände kodieren. Dadurch wird eine minimale Anzahl von Flipflops verwendet, so dass wiederum geringe Kosten erreicht werden können.

Bei der Gray-Kodierung (auch einschrittige Kodierung genannt) werden den Zuständen Binärwerte derart zugeordnet, dass bei einem Zustandswechsel nur ein Flipflop seinen Wert ändert. Dadurch wird für die Setz- und Rücksetzvorgänge der Flipflops nur wenig Hardware benötigt.

Bei der One-Hot-Kodierung wird jeder Zustand durch ein Flipflop repräsentiert. Dadurch wird die Schaltung zunächst teuer, da viele Flipflops verwendet werden, aber man erwartet, dass für die Auswertung der Zustandsübergänge einfachere Schaltnetze ausreichen, da nur einzelne Zustandsbits ausgewertet werden müssen. Letzlich soll der Flächenaufwand stets minimal sein. Diese Kodierung bietet sich für FPGAs an, da viele Flipflops zur Verfügung stehen.

Für sehr wenige Zustände ist es möglich, sämtliche Kodierungen auszuprobieren, die Kosten der Realisierungen zu bestimmen und die günstigste auszuwählen. Für größere Automaten ist dies nicht mehr möglich. Dann werden heuristische Verfahren verwendet, um eine möglichst kostengünstige Kodierung zu finden.

Register-Transfer-Synthese: FlipFlop-Auswahl und Schaltnetzsynthese



Nach der Zustandskodierung folgt die Auswahl der Flipflops (D-Flipflops sind hierfür am verbreitetsten). Die Steuereingänge der Flipflops bestimmen maßgeblich die aufzustellende Wahrheitstabelle für das Übergangsschaltnetz (Zustandsfolgetabelle). Die Wahrheitsstabelle des Ausgangsschaltnetzes (Ausgangstabelle) ist bereits durch die Zustandskodierung fest definiert. Die booleschen Gleichungen können unmittelbar aus den Tabellen abgelesen werden.