Definition 1 (Codes) Codes sind Vorschriften zur Abbildung eines Zeichenvorrates in einem anderen Zeichenvorrat.

Definition 2 (Zeichenvorrat) Ein Zeichenvorrat, auch Alphabet genannt,
ist die Menge der bei einem Code zur Verfügung stehenden Symbole.

Definition 3 (Binärcode) Ein Binärcode ist ein Code dessen Zeichenvorrat aus binären Werten besteht.

Definition 4 (Hammingdistanz) Die Hammingdistanz HD(c1, c2) zwi schen zwei gleich langen Codewörtern c1 und c2 ist die Anzahl der unterschiedlichen Binärstellen.

Beispiel: HD(1110, 0111) = 2 und HD((00101, 10110) = 3

Definition 5 (Boolesche Algebra) Eine Boolesche Algebra (B, ∨, ∧, ¬)
besteht aus:
(∨ heißt Disjunktion, Vereinigung, ODER, OR,...)
(∧ heißt Konjunktion, Durchschnitt, UND, AND,...)
(¬ heißt Negation, Komplement, NICHT, ...)

Definition 7 (Schaltagebra) Eine Boolesche Algebra mit kleinster Trägermenge {0, 1} nennt man eine Schaltalgebra.

Definition 8 (De Morgansche's Gesetz) Für eine Boolesche Algebra (B, ∨, ∧, ¬) gelten die Morganschen Gesetze.

Definition 9 (Schaltnetz) *Eine logische Schaltung (Abbildung 4.1) mit n Eingängen $x_1, x_2, \cdots, x_n$, die zum Tupel $X = (x_1, x_2, ..., x_n)$ gefasst werden können, und $m \geq 1$ Ausgängen $y_1, y_2 \cdots, y_m$, welche zum Funktionstupel $F(X) = (y_1 = f_1(X), y_2 = f_2(X), \cdots, y_m = f_m(X))$ zusammengestzt werden können, heißt **kombinatorische Schaltung** oder **Schaltnetz**, falls $F(X)$ zu jedem Zeitpunkt t ausschließlich durch X zum Zeitpunkt t bestimmt ist.*

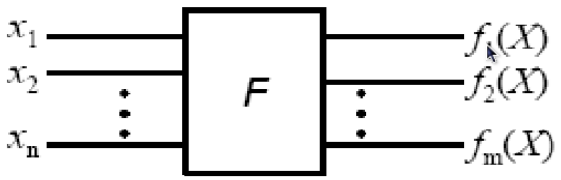


Abbildung 4.1: Schaltnetz als Blackbox mit Ausgängen als Funktion der Eingänge

Definition 10 (Synthese) Mit der Synthese bezeichnen wir den Entwurf
digitaler Schaltungen.

Wie oben erwähnt, ist die Hauptaufgabe der Synthese die optimale Umsetzung einer logischen Beschreibung in einer Schaltung mit Hilfe vorhandener Glieder. Im Gegenteil zur Synthese haben wir die Analyse, bei der versucht wird die logische Beschreibung der Funktion einer Schaltung

herauszufinden. Die Synthese digitaler Schaltungen wird im Laufe dieser Veranstaltung in den folgenden Schritten durchgeführt:

1. Ermittlung von Ein- und Ausgabe.
2. Ermittlung der Booleschen Gleichungen:
   Schritten:

(a) Erstellen der Wahrheitstabelle.
(b) Vereinfachung der logischen Verknüpfung und Darstellung mit den vorhandenen Gliedern.

3. Zusammensetzung der Schaltung.

Definition 11 (Literal) Mit Literal bezeichnen wir eine Boolesche Variable oder das Komplement einer Booleschen Variable.

Definition 12 (Produktterm) Ein Produktterm ist eine UND-Verknüpfung von $k \geq 1$ Literalen.

Definition 13 (Summenterm) Ein Summenterm ist eine ODER- Verknüpfung von $k \geq 1$ Literalen.

Definition 14 (Minterm) Ein Minterm oder eine Vollkonjunktion ist ein Produktterm in dem alle Variablen, negiert oder unnegiert, genau ein Mal vorkommen.

Definition 15 (Maxterm) Ein Maxterm oder Volldisjunktion ist ein Summenterm in dem alle Variablen, negiert oder unnegiert, genau ein Mal vorkommen.

Theorem 2 (disjunktive kanonische Form (DKF)) Jede Boolesche Funktion lässt sich eindeutig durch eine ODER-Verknüpfung (Disjunktion) von Mintermen darstellen.
Theorem 3 (Disjunktive Normalform (DNF)) Jede Schaltfunktion lässt sich durch eine ODER-Verknüpfung (Disjunktion) von Produkttermen darstellen.

Definition 16 (Karnaugh-Diagramm) Ein Karnaugh-Diagramm ist ein 2-dimensionales Feld rechteckiger Zellen zur Darstellung Boolescher Funktionen. Zur Darstellung einer Funktion von n Variablen sind 2^n Zellen notwendig

Definition 17 (Implikant) Ein Implikant ist ein Produktterm, bei dem der Wert/die Werte der Funktion in der Wahrheitstabelle oder in dem Karnaugh-Diagramm 1 ist/sind.

Definition 18 (Primimplikant) Ein Primimplikant ist ein Implikant, der keinen anderen Implikanten der Funktion umfasst.

Sum of Products (SOP) =DNF

Definition 19 (Essentieller Primimplikant) Ein essentieller Primimplikant oder Kern-Primimplikant ist ein Primimplikant, für den es einen Minterm gibt, der nur von diesem Primimplikanten überdeckt wird.

Definition 20 (Überdeckungstabelle) Eine Überdeckungstabelle ist eine zweidimensionale Tabelle, deren Zeilen die Primiplikanten und deren Spalten die Minterme darstellen. Falls ein Minterm m_i von einem Primimplikanten P_j überdeckt wird (der Minterm wurde bei der Ermittlung des Primimplikanten verwendet) wird ein X in der Tabelle an der Adresse (i, j) eingefügt.

Definition 21 (Zeilendominanz) Zeile P_i dominiert Zeile P_j , wenn Pi ein X in jeder Spalte hat, in der auch P_j ein X hat.

Definition 22 (Spaltendominanz) Spalte m_i dominiert Spalte m_j , wenn m_i ein X in jeder Zeile hat, in der auch m_j ein X hat.

Definition 23 (Mealy Automat)

$$EinMealyAutomatistein6-TupelA=(X,Y,S,\delta,\lambda,S_i),$$
$$wobei:$$
$$XeineendlichenichtleereMengevonSymbolen(dasEingabealphabet)ist,$$
$$YeineendlichenichtleereMengevonSymbolen(dasAusgabealphabet)ist,$$
$$SeineendlichenichtleereMengevonZuständenist,$$
$$\delta: S\times X\rightarrow SdieZustandsübergangsfunktionist,$$
$$\lambda: S\times X\rightarrow YdieAusgabefunktionist,und$$
$$S_iderInitialzustandist.$$

Definition 24 (Moore Automat)

$$EinMealyAutomatistein6-TupelA=(X,Y,S,\delta,\lambda,S_i),$$
$$wobei:$$
$$XeineendlichenichtleereMengevonSymbolen(dasEingabealphabet)ist,$$
$$YeineendlichenichtleereMengevonSymbolen(dasAusgabealphabet)ist,$$
$$SeineendlichenichtleereMengevonZuständenist,$$
$$\delta: S\times X\rightarrow SdieZustandsübergangsfunktionist,$$
$$\lambda: S\rightarrow YdieAusgabefunktionist,und$$
$$S_iderInitialzustandist.$$

Definition 25 (Wortfunktion) *Gegeben sei ein Automat* $A=(X,Y,S,\delta,\lambda,S_I)$. *Wir definieren zwei Wortfunktionen*
$\delta^*: S\times X^*\longrightarrow S^*$ *und*
$\lambda^*: S\times X^*\longrightarrow Y^*$ *wie folgt:*
$\forall i\in\{1\cdots n\}$:
$\delta(S_0,x_1,x_2,\cdots,x_n)=S_1S_2\cdots S_n$, *mit* $\delta(S_{i-1},x_i)=S_i$
$\lambda(S_0,x_1,x_2,\cdots,x_n)=y_1y_2\cdots y_n$, *mit* $\lambda(S_{i-1},x_i)=y_i$.

Mit den Wortfunktionen können wir jetzt die Äquivalenz zwischen Zuständen und Automaten definieren.

Definition 26 (Zustandsäquivalenz) *Gegeben sei ein Automat* $A=(X,Y,S,\delta,\lambda,S_I)$. *Zwei Zustände* S_1 *und* S_2 *heißen äquivalent* $(S_1\equiv S_2)$, *falls gilt:* $(S_1\equiv S_2)\Longleftrightarrow\forall x^*\in X^*,\lambda^*(S_1,x^*)=\lambda^*(S_2,x^*)$.

Definition 27 (k-Zustandsäquivalenz) *Gegeben sei ein Automat* $A=(X,Y,S,\delta,\lambda,S_I)$. *Zwei Zustände* S_1 *und* S_2 *heißen* k*-äquivalent* $(S_1\equiv_k S_2)$, *falls gilt:*
$(S_1\equiv_k S_2)\Rightarrow\forall x^*\in X^*l(x^*)\leq k^1,\ \lambda^*(S_1,x^*)=\lambda^*(S_2,x^*)$.

Theorem 4 *Gegeben sei ein Automat* $A=(X,Y,S,\delta,\lambda,S_I)$. *Sind zwei Zustände,* S_1 *und* S_2, *äquivalent, dann gilt:*

1. $\forall x\in X,\lambda(S_1,x)=\lambda(S_2,x)$
2. $\forall x\in X,\delta(S_1,x)\equiv\delta(S_2,x)$

Definition 28 (Automatenäquivalenz) *Zwei Automaten* $A = (X, Y, S, \delta, \lambda, S_I)$ *und* $A' = (X', Y', S', \delta', \lambda', S'_I)$ *sind äquivalent falls gilt:*
$\forall s \in S, \exists s' \in S'$ *mit* $s \equiv s'$ *und*
$\forall s' \in S', \exists s \in S$ *mit* $s' \equiv s$.

Definition 29 (Vollreduzierung) *Ein Automat* $A = (X, Y, S, \delta, \lambda, S_I)$ *heißt vollreduziert, falls gilt:* $\forall s_1, s_2 \in S : s_1 \equiv s_2 \Rightarrow s_1 = s_2$

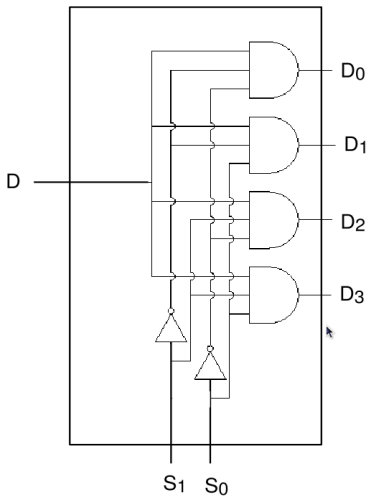


Abbildung 6.2: Schaltnetz eines 1:4 Demultiplexers

E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Tabelle 6.1: Wahrheitstabelle eines Binär-Encoder mit 8 Eingängen und 3 Ausgängen

A_i	C_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabelle 6.3: Wahrheitstabelle eines Volladdierers $S_i = A_i \oplus B_i \oplus C_{i-1}$, $C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$ (siehe Abbildung 6.2)

Ein $n : 1$ (n zu 1) **Multiplexer** ist eine kombinatorische Einheit mit n Dateneingängen $\{D_0, \cdots, D_{n-1}\}$, k Selektoreingängen $\{S_0, \cdots, S_{k-1}\}$ und einem Ausgang (Abbildung 6.1).

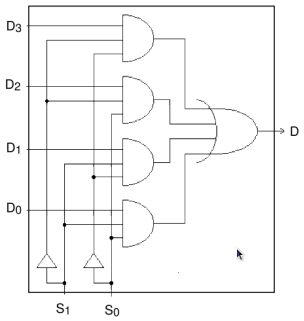


Abbildung 6.1: Schaltnetz eines 4:1 Multiplexers

E_2	E_1	E_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Tabelle 6.2: Wahrheitstabelle eines Binär-Decoders mit 3 Eingängen und 8 Ausgängen

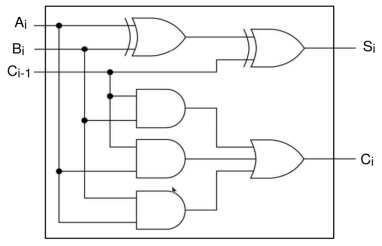


Abbildung 6.3: Schaltnetz eines Volladdierers

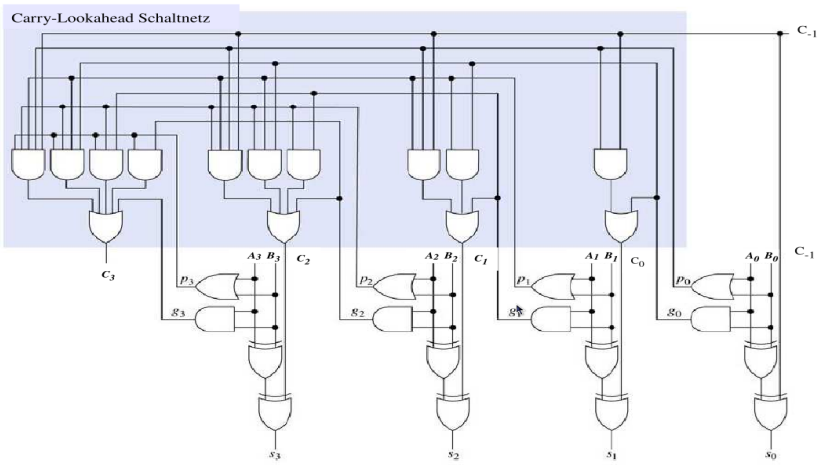
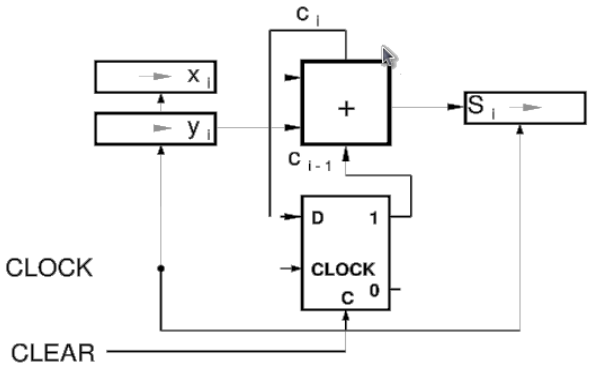
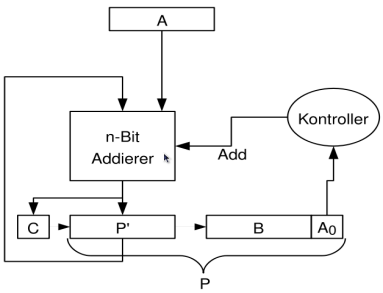
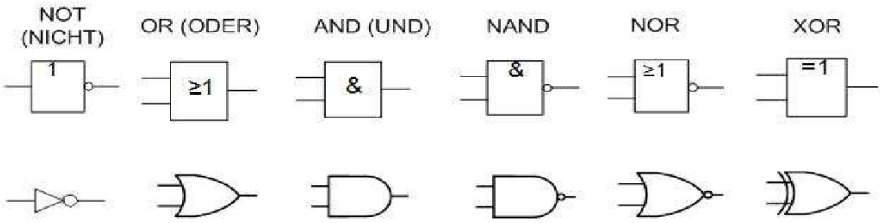


Abbildung 6.5: Schaltnetz eines 4-Bit Carry-Lookahead-Addierers.

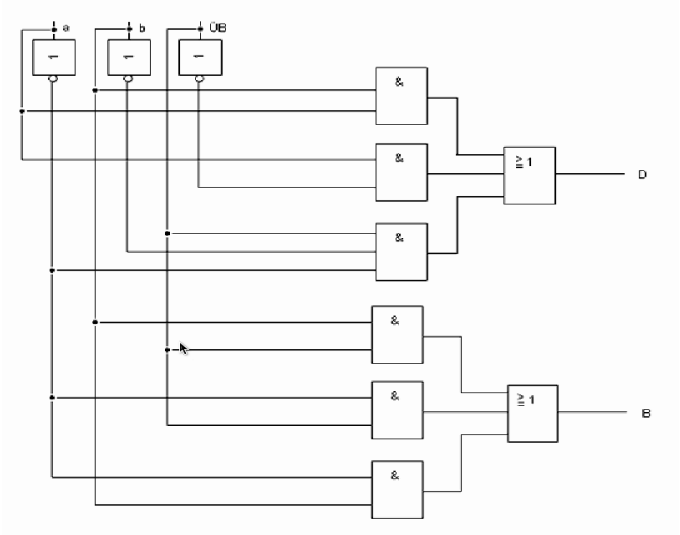




.8: Blockschaltbild eines n-Bit-"Shift & Add"-Multiplizierers.



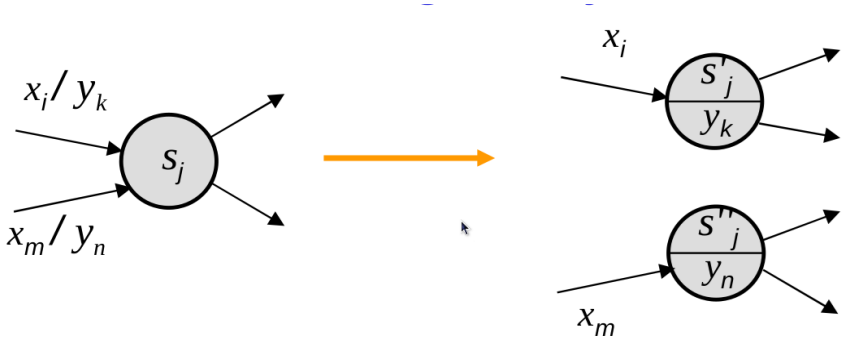
Vollsubtrahierer



Zustandsminimierung:
1. Sortieren nach Ausgabeklassen

2. Partitioniere weiter jede Äquivalenzklasse nach den Folgezustandsklassen in die sie für jede Eingabe übergehen.

3. Falls keine Äquivalenzklasse mehr partitioniert werden kann, terminiert das Verfahren. Alle ermittelten Äquivalenzklassen bilden die neuen Zustände des minimalen Automaten. Füge die Zustandsübergänge im neuen Automaten hinzu. Ein Übergang wird zwischen zwei Klassen hinzugefügt, falls zwei Elemente (Zustände) der beiden Äquivalenzklassen einen Übergang im ursprünglichen Automaten besitzen.
Mealy zu moore:

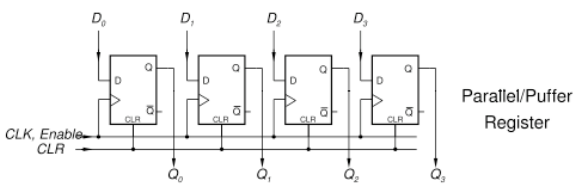


Welcher algorithmus für divison? Was beachten?

Restoring-Algorithmus ; man muss beachten, ob das ergebnis positiv oder negativ ist

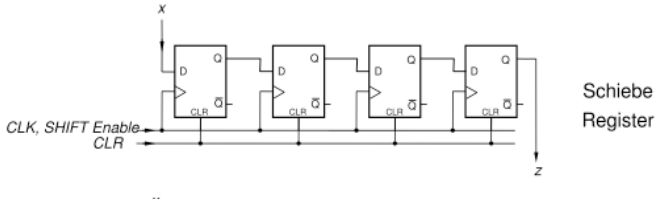
Ein Register ist ein kleiner Speicher, der meisten nur für die Speicherung eines Wortes vorgesehen ist:

- Ein n-Bit Parallelregister, auch als Pufferregister bekannt, besteht aus n verschiedenen Flip-Flops, die parallel beschrieben und parallel ausgelesen werden können.

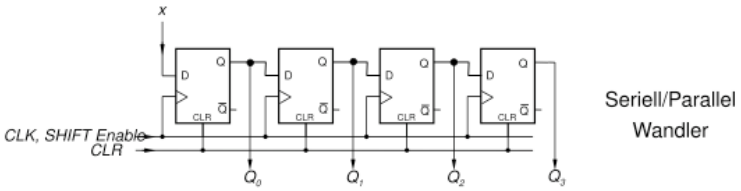


- Ein n-Bit Schieberegister ist eine Kaskadierung von n Flip-Flops. Der Eingang des Schieberegisters ist der Eingang des ersten Registers in der Kette und sein Ausgang der des letzten Registers in der Kette. Jedes

Register in der Kette bekommt seinen Eingang vom Ausgang des vorherigen Registers in der Kette und sein Ausgang wird als Eingang des nächsten Registers in der Kette benutzt. Jedes Bit muss durch die n Rgister in der Kette gehen, um am Ende zu erscheinen.

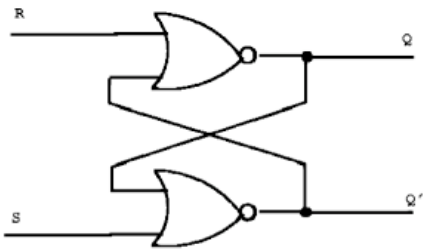


- Seriell-Parallel-Wandler: In vielen Anwendungen werden die Daten bitweise empfangen. Zur Verarbeitung der Daten in Recheneinheiten, die meist auf breiteren Worten arbeiten, muss eine Konvertierung erfolgen. Mit einem Seriell-Parallel-Wandler ist es möglich das Register seriell zu beschreiben und parallel auszulesen. Nach n Schritten enthält das Reigster ein n-Bit-Wort, das der Recheneinheit bereit gestellt werden kann. Umgekehrt stellen Parallel-Seriell-Wandler parallel empfangene Daten bitweise bereit, z.B. für die Übertragung. Das parallele Laden der Daten erfolgt unter der Kontrolle einer separaten Leitung LOAD.



Flip-Flop (auch Bistabile genannt) ist ein kombinatorisches Speicherelement, das zwei Zustände (0 und 1) für beliebig lange Zeit (solange Strom vorhanden ist) annehmen kann.

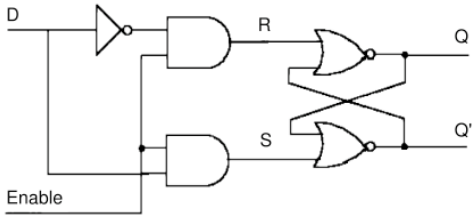
Ein RS-Flip-Flop ist eine Flip-Flop-Variante mit zwei Eingängen, S (set) und R (reset), und zwei komplementären Ausgängen Q und Q'. Ein RS-Flip-Flop wird durch zwei NOR-Gatter implementiert, deren zweite Eingänge die Ausgänge des anderen NOR-Gatters sind.



R	S	Q_{neue}	Q'_{neue}	Anmerkung
0	0	Q_{alt}	Q'_{alt}	Halten
0	1	1	0	Setzen
1	0	0	1	Rücksetzen
1	1	0	0	nicht zulässig
0	0	?	?	undefiniert

5.6.2 D-Flip-Flops

Eine Anomalie des RS-Flip-Flops kann durch die Vermeidung der ungültigen Eingangskombination korrigiert werden. Das resultierende Element heißt **D-Flip-Flop** (Delay-Flip-Flop).
Ein D-Flip-Flop hat nur einen Dateneingang und einen Kontrolleingang. Dieser bestimmt, wann Daten in den Flip-Flop kopiert werden. Wenn die Enable-Leitung den Wert 1 annimmt, erscheint der Wert am Eingang D und am Ausgang Q. Man spricht in diesem Fall von einem **transparenten Zustand**, in dem sich das Flip-Flop befindet.
Der Begriff Flip-Flop wird meist für bistabile Elemente verwendet, die sich nur für kurze Zeit im transparentem Zustand befinden können und somit den stabilen Betrieb des Systems, in das sie integriert werden, sichern. Für Bistabile, die sich *beliebig lange im transparenten Zustand* befinden können, ohne den Betrieb des System, in dem sie arbeiten, zu beeinträchtigen, wird öfter der Begriff **Latch** verwendet. D-Flip-Flop entspricht ca. D-Latch



Der Entwurf kombinatorischer Schaltungen erfolgt in 3 Schritten:

Modellierung: Zuerst muss festgelegt werden, was die Eingabe und Ausgabe der Einheit ist. Der Ausgang bestimmt die Funktion der Schaltung. Das Modell legt fest, wie die Eingabe und Ausgabe der Einheit aussieht und bietet Optimierungsmöglichkeiten an.

Synthese: Bei der Synthese geht es darum, die optimale Schaltung mit den vorhandenen Gliedern zu ermitteln, sodass gegebene Randbedingungen erfüllt werden. Es muss dabei die minimale Schaltung (in Form von Gliedern) berechnet werden, um die kleinste Chipfläche zu realisieren, da wenig verwendete Glieder wenig Fläche bedeuten. Außerdem muss erreicht werden, dass die Schaltung nicht zu hoch getaktet wird, da dies mehr Stromverbrauch bedeuten würde. Die Synthese ist eine sehr komplexe Aufgabe, vor allem, weil die Optimierungsziele meistens widersprüchlich zueinander sind.

Implementierung: Die Implementierung realisiert letztendlich die Schaltung in einer gewissen Technologie. Chips werden heute in Halbleitertechnologie realisiert.

Schaltnetz – eine logische Schaltung mit n Eingängen $x_1, \ldots, x_n$, die zum Tupel $X = (x_1, \ldots, x_n)$ gefasst werden können, und $m \geq 1$ Ausgängen $y_1, y_2, \ldots, y_m$, welche zum Funktionstupel
$F(X) = (y_1 = f_1(X), \ldots, y_m = f_m(X))$ zusammengesetzt werden können, heißt *kombinatorische Schaltung* oder *Schaltnetz*, falls F(X) zu jedem Zeitpunkt t ausschließlich durch X zum Zeitpunkt t bestimmt ist.

Synthese – bezeichnet den Entwurf digitaler Schaltungen.

Wie oben erwähnt ist die Hauptaufgabe der Synthese die optimale Umsetzung einer logischen Beschreibung in einer Schaltung mit Hilfe vorhandener Glieder. Im Gegenteil zur Synthese haben wir die **Analyse**, bei der versucht wird die logische Beschreibung der Funktion einer Schaltung herauszufinden. Die Synthese digitaler Schaltungen wird in drei Schritten realisiert:

1. Ermittlung von Ein- und Ausgabe: Aus der Spezifikation ermittelt, wie viele Ein- und Ausgänge die Schaltung haben soll. Bei der Ermittlung der Ein- und Ausgangsvariablen muss die Bedeutung von 0 und 1 festgelegt werden.
2. Ermittlung der Booleschen Gleichungen: Jede Ausgangsvariable (Funktion) wird als Boolesche Funktion der Eingänge definiert. Dies geschieht in zwei Schritten:

a) Erstellen der Wahrheitstabelle für jede Ausgangsvariable (Funktion) und Bestimmung der logischen Verknüpfung für jede einzelne Funktion.
b) Vereinfachung der logischen Verknüpfung und Darstellung mit den vorhandenen Gliedern.

3. Zusammensetzung der Schaltung.