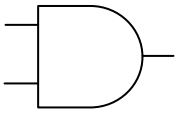
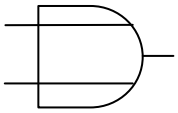
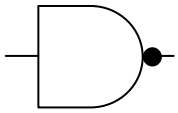
Rechnerstrukturen

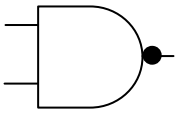
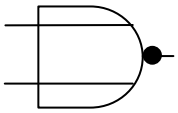
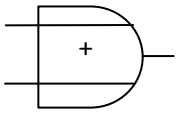
Zahlendarstellungen:

Festkommazahlen:

$01101.111_2 \rightarrow 13.875\ (2^3+2^2+1+2^{-1}+2^{-2}+2^{-3})$

Gatter:

AND :  OR:  NOT: 

NAND:  NOR:  XOR: 

Halb- & Volladdierer:

Halbaddierer: $S = A \oplus B \;\; CO = A * B$

Volladdierer: $S = A \oplus B \oplus CI \;\; CO = AB + ACI + BCI$

Ripple-Addierer: Aneinanderreihung von Volladdierern, die jeweis eine Stelle ausrechnen.
A0B0 -> S0 etc.

Carry-Select-Addierer: Rechnen mit beiden Carries -> Auswahl per Multiplexer

Tabellenbasierter Addierer: Vorberechnete Summen gespeichert

Carry-Lookahead Addierer:

Schaltnetze:

Schaltung aus Gattern mit n Eingängen, m Ausgängen und ohne Rückkopplung

Es gibt also 2^n Eingangskombinationen und m boolsche Ausdrücke

Angabe der 1er Zeilen in der Wahrheitstabelle: $\sum\{1,2,5,7\}$, 0er Zeilen: $\prod\{0,3,4,6\}$

Disjunktive Normalform: Summe von Produkten (minterme)

Konjunktive Normalform: Produkt von Summen (maxterme)

Minterm: 1 Zeile in der WHT (Wahrheitstabelle), 1 -> e, 0 -> $\bar{e}$

Maxterm: 0 Zeile in der WHT, 1 -> $\bar{e}$, 0 -> e

Anwendung: Umwandler (En-/Decoder), Verteiler (Multiplexer) etc

1 aus n Decoder (zum Auswählen/Aktivieren etc)

1 aus n Encoder (Prioritätsdecoder) wandelt n Leitungen in bin. Zahl um

Paritätsdecoder (Gerade/Ungerade Test von n Eingängen)

Hazards:

Statischer 1 Hazard: Kurzzeitig ungewollt von 1 auf 0

Statischer 2 Hazard: Kurzzeitig ungewollt von 0 auf 1

PLA/PAL:

PLA: Programmierbare UND- oder ODER-Gatter

PAL: Programmierbare UND-Gatter

Schaltwerke:

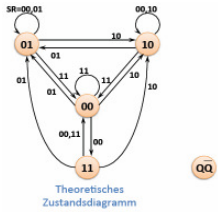
Ausgang hängt von Eingängen und vorherigen Ausgängen ab

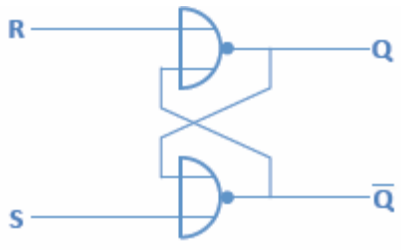
Ermöglicht Speicherung

RS-Latch:

S: Set

R: Reset

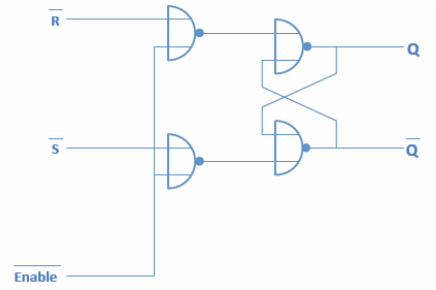




Kreuzverschaltete NOR-Gatter

Steuerungsarten:

Pegelgesteuert: Bei 1 aktiv und Änderung möglich →



Flankengesteuert (Positiv): Bei Flanke von 0 auf 1 aktiv

Flankengesteuert (Negativ): Bei Flanke von 1 auf 0 aktiv

Genutzte Steuerungsleitungen: Enable, Clock

Latch:

Pegelgesteuert, Ungesteuert

Änderung der Ausgänge bei Änderung der Eingänge

Flip-Flop:

Flankengesteuert

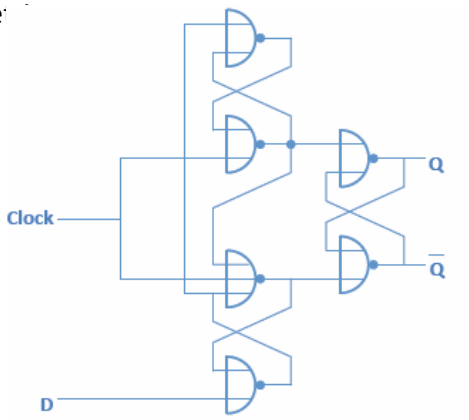
Master-Slave: Realisierung der Flankensteuerung. Erstes Latch speichert immer, zweites übernimmt bei Übergang auf 0/1

Änderung der Ausgänge wird durch Steuerungseingang ge

D-Flip-Flop:

Reine Flankensteuerung: Keine ungültigen Eingaben

Negativ Flankengesteuert



Minimierung:

Verfahren:

Karnaugh-Veit-Diagramm (KV)

DNF durch 1en, KNF durch 0en

Bündelminimierung

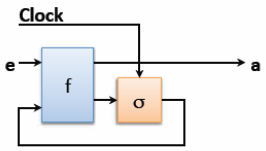
Automaten:

Umsetzung von Zustandsdiagrammen

Zustand: Bitvektor (Register)

Übergangsfunktion: Schaltnetz

Mealy-Automat:



Asynchrone Ausgabe (Lösung durch Taktung)

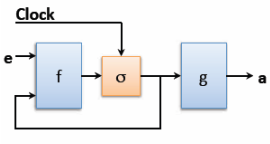
Ausgaben werden durch Zustand und Eingaben bestimmt

Ausgaben werden im Zustandsdiagramm neben Übergängen geschrieben

Benötigt meißt weniger Zustände

$$FSM = (\Theta, \sigma_0, f, E, A)$$
$$f: \Theta \times E \to \Theta \times A$$
$$(\sigma', a) = f(\sigma, e)$$

Moore-Automat:



Synchrone Ausgabe

Ausgaben nur durch Zustand bestimmt

Ausgabe neben Zustände geschrieben

$$FSM = (\Theta, \sigma_0, f, g, E, A)$$
$$f: \Theta \times E \to \Theta, \quad \sigma' = f(\sigma, e)$$
$$g: \Theta \to A, \quad a = g(\sigma)$$

Übergangsfunktion:

- Beschreibung durch Wertetabelle
- Auch Zustände lassen sich minimieren
- Flip-Flop-Typ wählen
- Ansteuerungsfunktion bestimmen und minimieren

Endliche Automaten:

- $FSM = (\Theta, \sigma_0, f, E, A), f: (\Theta \times E \to \Theta \times A)\ (\sigma', a) = f(\sigma, e)$
- Endliche zustandsmenge Θ
- Initialzustand σ_0
- Übergangsfunktion f
- Ein- und Ausgabealphabet E,A