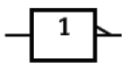
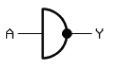
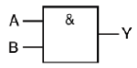
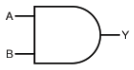
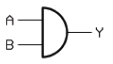
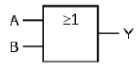
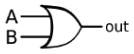
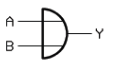
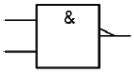
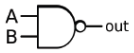
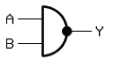
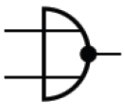
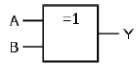
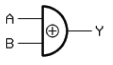
5. Kapitel: Schaltnetze I

- Y = F(X)
- X: Eingangsvektor (mit den Eingangsvariablen)
- Y: Ausgangsvektor (mit den Ausgangsvariablen)
- F: Funktion (gebildet durch elementare Operatoren und, oder, nicht)
- Signallaufzeiten werden bei der Beschreibung nicht berücksichtigt
- Ein Schaltnetz ist ein Schaltwerk, dessen Wert am Ausgang zu einem beliebigen Zeitpunkt nur von den Eingangswerten zu diesem Zeitpunkt abhängt.
- Analyse: Überführung eines Schaltplans in ein Schaltnetz (Gleichung, Tabelle erstellen)
- Synthese: Überführung eines Schaltnetzes in einen Schaltplan
- Arten:
    - Festverdrahtete Logik (Aufbau durch einzelne Verknüpfungsglieder und, oder nicht)
    - Adressgesteuerte Logik (Multiplexer, (P)ROM)
- 4 Huntigton'sche Axiome aus denen man alle weiteren Gesetze ableiten kann:
    - Kommutativ-, Assoziativ-, Neutralitäts-, Dualitätsgesetz

Operator	Schaltzeichen (IEC)	(ANSI)	(DIN/alt)	Benennung	
$\neg$				Nicht	Not
$\wedge$				Und	And
$\vee$				Oder	Or
$\overline{\wedge}$				Und nicht	Nand
$\overline{\vee}$				Oder nicht	Nor
$\oplus/\underline{\vee}$				Exklusiv-Oder	Xor

- Bei n Varibalen → 2^n Wertekombinationen
- Minterm (und): (a ∧ b ∧ c ∧ d) → Disjunktive Normalform (DNF)
- Maxterm (oder): (a v b v c v d) → Konjunktive Normalform (KNF)
- Verfahren zur Minimierung: (Grundlage aller das Komplementärgesetz)
    - Gesetze der Schaltalgebra
    - Karnaugh/Veitch - Diagramme
    - Quine/McCluskey - Methode
- KV: ab 6 Variablen zu unübersichtlich
- QMC: ab 15-20 Variablen zu uneffektiv
- Daher bei hoher Variablenanzahl werden heuristische Methoden angewendet. Damit kommt man nicht zum Besten aber zu einem akzeptablen Ergebnis.

	x_4	x_3	x_2	x_1		Minterme
(2,10)	-	0	1	0	→	$\neg x_3 \wedge x_2 \wedge \neg x_1$
(10,11)	1	0	1	-		$x_4 \wedge \neg x_3 \wedge x_2$
(0,2,4,6)	0	-	-	0		$\neg x_4 \wedge \neg x_1$
(4,5,6,7)	0	1	-	-		$\neg x_4 \wedge x_3$

6. Kapitel: Schaltnetze II

- Vollständige Operatorensysteme (AND/OR/NOT, NAND, NOR)
- Technische Realisierung mit NAND und NOR oft einfacher
- Minimierung von Vektorfunktionen
  - KV für jede y-Komponente
  - Besser: Bündelminimierung mit QMC

DNF → NOR

$$y = \bar{a}bc \vee a\bar{b}c \vee ab\bar{c} \vee \bar{a}\bar{b}\bar{c}$$

$$\bar{y} = \overline{\bar{a}bc \vee a\bar{b}c \vee ab\bar{c} \vee \bar{a}\bar{b}\bar{c}} \quad (1)$$

$$= \overline{\bar{a}bc} \vee \overline{a\bar{b}c} \vee \overline{ab\bar{c}} \vee \overline{\bar{a}\bar{b}\bar{c}} \quad (2)$$

$$= \overline{(a \vee \bar{b} \vee \bar{c})} \vee \overline{(\bar{a} \vee b \vee c)} \vee \overline{(\bar{a} \vee \bar{b} \vee c)} \vee \overline{(a \vee b \vee c)} \quad (2)$$

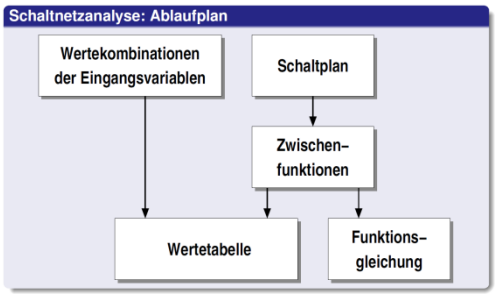
$\bar{y} \rightarrow y$ per Negierung: $\bar{y} = \overline{y \vee y}$ (3)

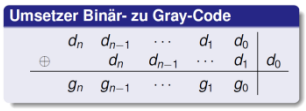
DNF → NAND

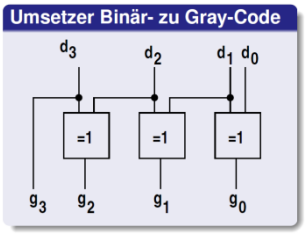
$$y = \bar{a}bc \vee a\bar{b}c \vee ab\bar{c} \vee \bar{a}\bar{b}\bar{c}$$

$$= \overline{\bar{a}bc \vee a\bar{b}c \vee ab\bar{c} \vee \bar{a}\bar{b}\bar{c}}$$

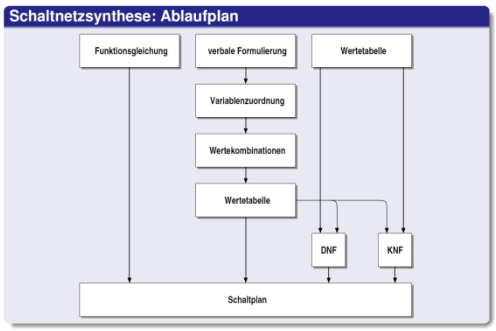
$$= \overline{\bar{a}bc} \wedge \overline{a\bar{b}c} \wedge \overline{ab\bar{c}} \wedge \overline{\bar{a}\bar{b}\bar{c}}$$



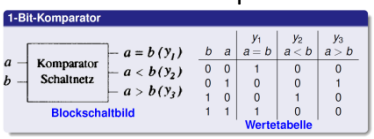




- **Gray-Code**: Bei Übergang von einer Zahl zur nächsten wechselt nur eine einzelne Bitstelle.
- Verwendet in der Messtechnik (z.B. Mausrad)
- **Adressdecodierer**: Anwahl von Bausteinen, etc. über Adresse (n Eingänge, 2^n Ausgänge)
- **Multiplexer** ist ein Auswahl-Schaltnetz. Steuereingänge (S_i) schalten bei vorliegender Freigabe ($\bar{E}$) einen von mehreren Dateneingängen (D_i) auf den Ausgang (Y) durch. (z.B. 4:1-Multiplexer)



- Anwendung von MUX: Jede UND-Verknüpfung der Steuer- mit den Dateneingängen hat adressierende Eigenschaften. Jede Kombination der S wird mit einem D UND-verknüpft und auf den Ausgang durchgeschaltet.
- **Demultiplexer**: Verteilendes Schaltnetz (n S-eingänge schalten D-eingang auf 2^n Ausgängen)

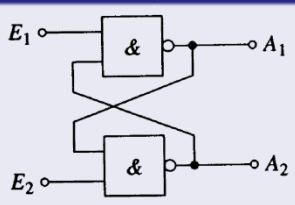


- Komparatoren: Vergleichen analoge oder binäre Signale (n-Bit-Komparator)
- Aufbau höherwertiger arithmetischer Funktionen mit den gezeigten Mitteln machbar!
  - Addition/Subtraktion per Wertetabelle aufwendig
  - Multiplikation/Division in diskreter Logik mit exponentiellem Flächenbedarf

7. Kapitel: Speicherglieder

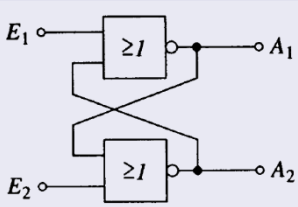
- Ausgangswert ist immer Funktion der Eingangswerte (es gibt kein Gedächtnis), darum werden Speicherglieder benötigt die temporär Signale speichern (Adressen, Daten)

FF aus NAND-Gattern ($\overline{RS}$-NAND-Latch)



$E_2(\bar{R})$	$E_1(\bar{S})$	$A_2(\bar{Q})$	$A_1(Q)$
0	0	speichern	
0	1	0	1
1	0	1	0
1	1	unzulässig	

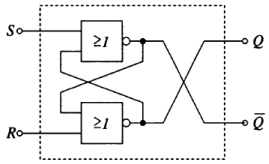
FF aus NOR-Gattern (RS-NOR-Latch)



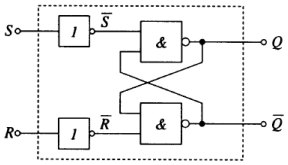
$E_2(S)$	$E_1(R)$	$A_2(\bar{Q})$	$A_1(Q)$
1	1	speichern	
1	0	0	1
0	1	1	0
0	0	unzulässig	

und den internen Zustand sichern. (Zähler, Ablaufsteuerung).
→ Bistabile Kippschaltung (Flip-Flop)

- Flip-Flops werden über Kreuzkopplung zweier Gatter realisiert. (→ Rückkopplung)
- **Taktabhängigkeit**
  - Taktunabhängige FF: Kippvorgang wird durch Änderung der Eingänge ausgelöst.
  - Taktabhängige FF: Übernahme der Eingänge und Auslösung des Kippvorgangs synchronisiert mit Taktsignal
    - Pegelgesteuert: ändert Zustand bei anliegendem 1-Pegel (oder 0-Pegel)
    - Flankengesteuert: ändert Zustand nur mit steigender [0 -> 1] (oder fallender [1 -> 0]) Taktflanke
- **Taktunabhängiges RS-Flipflop**
  - Grund-FF mit Änderung der Ein- und Ausgänge
  - Charakteristische Schaltfunktion
    - $Q_{n+1} = S \vee (\neg R \wedge Q_n)$
- **Taktzustandgesteuertes RS-Flipflop**
  - Taktsignal sperrt/öffnet die FF-Eingänge
  - Sicherstellung, dass Eingänge erst geöffnet werden, wenn die anliegende Signale Stabil sind
- **Taktzustandgesteuertes D-Flipflop**
  - Ableitung aus RS-FF über D = S = ¬R
    → vermeidet unzulässige Eingangskombination
- **Taktzustandgesteuertes JK-Flipflop**
  - Verallgemeinerung von D- und T-FF
- **Taktzustandgesteuertes T-Flipflop**
  - Ableitung aus FK-FF über T = J = K
    → nur noch speichern oder kippen
  - Ableitung aus D-FF über
    $T = (D \wedge Q_n) \vee (D \wedge Q_n)$



RS-FF (NOR)

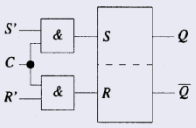


RS-FF (NAND)

Taktzustandsgesteuertes RS-Flipflop (forts.)

- Zustandsfolgetabelle

C	S′	R′	Q_{n+1}	
0	*	*	Q_n	Speichern (implizit)
1	0	0	Q_n	Speichern (explizit)
1	0	1	0	Rücksetzen
1	1	0	1	Setzen
1	1	1	X	unzulässig



Schaltung

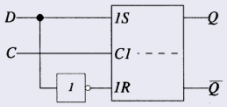
- Charakteristische Gleichung

$Q_{n+1} = (S' \wedge C) \vee (Q_n \wedge \overline{R'}) \vee (Q_n \wedge \overline{C})$
$= (S' \wedge C) \vee (Q_n \wedge (\overline{R' \wedge C}))$
intern: $S = S' \wedge C$ und $R = R' \wedge C$

Taktzustandsgesteuertes D-Flipflop

- Zustandsfolgetabelle

D	S	R	Q_{n+1}
0	0	1	0
1	1	0	1



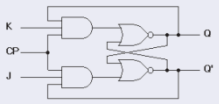
Schaltung

- Charakteristische Gleichung
  - $Q_{n+1} = D$

Taktzustandsgesteuertes JK-Flipflop

- Zustandsfolgetabelle

J	K	Q_{n+1}	
0	0	Q_n	speichern
0	1	0	löschen (kill)
1	0	1	setzen (jump)
1	1	$\overline{Q_n}$	invertieren (toggle)



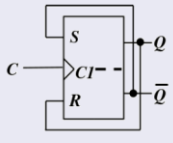
Q_n	Q_{n+1}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

- Charakteristische Gleichung
  - $Q_{n+1} = (J \wedge \overline{Q_n}) \vee (\overline{K} \wedge Q_n)$
- JK-FF ist Verallgemeinerung
  - D-FF: $K = \overline{J}$
  - T-FF: $K = J$

Taktzustandsgesteuertes T-Flipflop

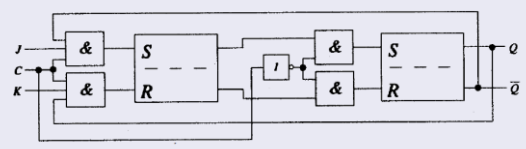
- Zustandsfolgetabelle

T	J	K	Q_{n+1}
0	0	0	Q_n
1	1	1	$\overline{Q_n}$



- Charakteristische Gleichung
  - $Q_{n+1} = \overline{Q_n}$

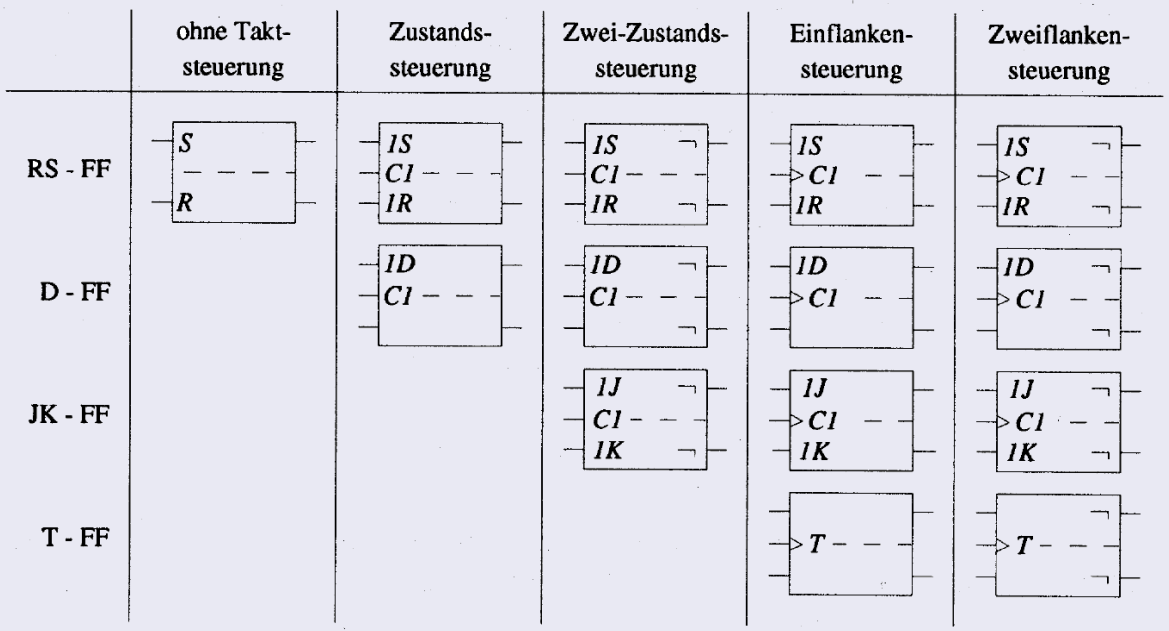
Taktzustandsgesteuertes JK-Master/Slave-FF



- **Master/Slave-FF**: hintereinander geschaltete FFs
  Bildet Zwischenzustände.
  - Master speichert mit positiven Taktsignal
  - Slave speichert mit invertiertem Taktsignal

- Schaltschwellen: Taktsignal verläuft nicht rechteckförmig, somit kommt es zu Fehlern und gegebenfalls kurzfristigen Freigaben beider FFs → Schaltschwelle für Inverter niedriger als Schaltschwelle beider Eingangs-UNDs
- **Taktflankensteuerung:** Taktflanke definiert genauen Zeitpunkt der Zustandsänderungen
- **Zweiflankensteuerung:**
  - Anwendung der Taktflankensteuerung auf Master/Slave-FFs
  - Eingänge der beiden FFs zu entgegengesetzten Taktflanken geöffnet

Überblick über gebräuchliche FFs



- **Hazard**: Unerwünschtes Auftreten kurzer Impulse am Ausgang der Schaltung
  - *Statischer Hazard*: Störimpuls einer Verknüpfung, die eigentlich konstant 0/1 liefert
  - *Dynamischer Hazard*: Zusätzliche Übergänge am Ausgang eines VK-Glieds
  - *Logik-/Strukturhazard*: Auftreten bei Übergang einer einzigen Eingangsgröße, Beseitigung durch Änderung der Schaltung
  - *Funktionshazard*: Auftreten beim gleichzeitigen Übergang mehrerer Eingangsgrößen, Ursache liegt in der Schaltfunktion selbst, Korrektur der Schaltfunktion erforderlich
- **Speicherglieder**: Grundelement dieser ist das Flipflop als elementare Speicherzelle.
- Mit dem Zusammenfassen von FF werden so Speicherworte 8, 16 … Bit möglich. Diese werden über eine Speicheradresse angesprochen.
- **Speichertechnologien**
  - Statischer Speicher (SRAM)
    - Speicherform über CMOS
  - Dynamischer Speicher (DRAM)
    - Speicherform über Kondensator
- **RAM: Random-Access Memory**
  - Adressleitungen: A_0, … A_{n-1} (Adressierungsbreite (2^n Speicherworte): n)
  - Datenleitungen: D_0, … D_{m-1} (Datenbreite m)
  - Chip Select (Bausteinauswahl): CS

- Lese-/Schreibanwahl: R/W (0 schreiben, 1 lesen)

8. Kapitel: Hardware

- **Programmierbare Logikbausteine**
  - „Freie“ Verschaltung der Logik-Ressourcne
  - Steigende Komplexität (PLDs, CPLDs, FPGAs)
  - Reprogrammierbare Bausteine erstmals rekonfigurierbare Systeme möglich
- **Begriffe**
  - PLDs: Programmable Logic Devices
  - S(imple)PLDs: Einfache Logikbausteine, basierund auf UND- und ODER Matrix zur Modellierung von Ein- und Ausgabeverhalten (PLA, PAL, GAL)
  - C(omplex)PLDs: SPLD-Weiterentwicklungen mit mehr als 1000 Gatteräquivalenten
  - FPGAs: Field-programmable Gate Arrays
    - Komplexe matrixstruktur aus konfigurierbaren Logikblöcken
  - ASCIs: Application-specific Integrated Circuits (ähnlich zu FPGA)

Klassifikation

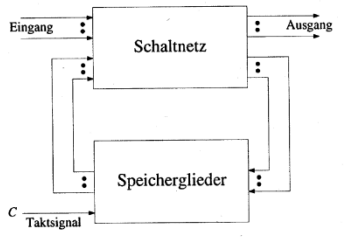
Typ		Matrix UND	Matrix ODER
PLA	Programmable Logic Array	prog.	prog.
PAL	Programmable Array Logic	prog.	fest
ROM	Read-only Memory	fest	fest
PROM	Programmable ROM	fest	prog.
EPROM	Erasable PROM	fest	prog.
EEPROM	Electrically EPROM	fest	prog.
FPGA	Field-programmable Gate-Array	n/a	

- **PLD**s sind aufgebaut aus dreistufigem Schaltnetz
- Resultierendes Schaltnetz realisiert Schaltfunktion in DNF
- **ROM** (Read-only Memory)
  - UND-Matrix vordefiniert, ODER-Matrix programmierbar
  - Anfänge: Maskenprogrammierung
    - Vorgabe einer Schaltmatrik, einfügen oder weglassen einer Verbindung
  - Feldprogrammierbarkeit durch Fuse-Technologie (PROM)
    - Auslieferung einer totalverbundenen Schaltmatrix, Programmierung, durch wegbrennen unbenötigter Verbindungen. Somit nur 1x programmierbar und ebenso nicht löschbar. Hohe Spannung und viel Zeit benötigt
- Reprogrammierbarkeit: **Floating Gates** (FG)
  - Programmieren des Gates über erhöhte Programmierspannung (ca. 12-25 Volt)
  - Löschen des Gates mittels UV-Lichts bestimmter Wellenlänge
- Speicherzelle mit Floating Gate (Verwendung von FG-MOSFETs)
  - FG ist isoliert und kann Ladung über ausgedehnten Zeitraum speichern
- FG – Lesen über anlegen einer Lesespannung
  - Keine Ladung auf Floating Gate (0) (Ausgang = 0 Volt)
  - Ladung auf Floating Gate (1) (Ausgang > 0 Volt)
- FG – Schreiben über Aufbringen von Ladung
  - Anlegen der Programmierspannung zwischen Drain und Source erzeugt Lawinendurchbruch in Raumladungszone, dadurch gelangen sie auf das FG
- FP – Löschen über Bestrahlung mit UV-Licht
  - UV-Licht bewirkt Bildung von Elektron/Loch-Paaren in Oxidschicht, die Löscher in der Oxidschicht werden von Elektronen auf FG angezogen und neutralisieren diese
- Interessant: Beim Absenken der Versorgungsspannung treten unzureichend gelöschte Daten wieder zutage.

- **Elektrische Reprogrammierbarkeit:**
  - Wiederprogrammierbare Festwertspeicher (ähnlich zu RAM, beschränkte Wiederbeschreibbarkeit, → EEPROM, FlashROM)
  - **Flash-Speicher**: Löschung mit geriner Granulairtät (z.B. 256-Byte-Sektoren), Anwendung als EPROM- und Festplattenersatz (On-board-Programmierung, Solid-state Disk, ggf. eingebaute Programmierhelfen und abstrakteres Interface)
    - **Vorteile**:
      - Datenerhalt bei fehlender Versorgungsspannung
      - Geringer Energieverbrauch im Betrieb
      - Günstiger Massenspeicher
      - Resistent gegen Erschütterungen und kleine Bauform
    - Nachteile:
      - Zugriff langsamer als RAM, insbes. Schreiben
      - Vergleichsweise komplexe Ansteuerung
      - Begrenzte Anzahl von Schreibzyklen
      - Löschgranularität sektorweise
- **Uniforme Bezeichung von PALs und GALs** PLDxxyyzz
  - PLD bezeichnet die Bausteinfamilie (PAL, PALCE, GAL)
  - xx die maximal mögliche Anzahl von Eingängen, welche zur Bildung von Produkttermen herangezogen werden können
  - zz Anzahl der Ausgänge
  - yy Ausgangskonfiguration (H: high-aktive Logik, L: low-akt L., R: Register im Ausgang)
- Zur Erhöhung der Bausteinkapazität, verwendet man **Multi-SPLD**
- **FPGA (Field Programmable Gate Array)** ist ein integrierter Schaltkreis der Digitaltechnik, in den eine logische Schaltung programmiert werden kann. Durch die spezifische Konfiguration interner Srukturen können in einem FPGA verschiedene Schaltungen realisiert werden. Diese reichen von einfachen Schaltungen (z.B. Synchronzähler), bis zu hochkomplexen Schaltungen (z.B. Mikroprozessor). Vorteile sind die schnelle Signalverarbeitung und die flexible Änderung der Schaltung, um z.B. nachträglich implementierte Funktionen verbessern zu können ohne Hardwareänderungen vorzunehmen zu müssen.

CPLD		FPGA
wenige große Logikblöcke, eingeschränkte, fest vorgegebene Routing-Matrix	**Struktur**	viele kleine Logikblöcke, aufwendige und flexible Routing-Matrix
zentral, global	**Verbindung**	dezentral, lokal
homogen, konstant und kurz, daher vorhersagbar		
nichtflüchtig (EEPROM)	**Konfiguration**	flüchtig (SRAM)
typ. <100.000 Gatteräquivalente	**Größe**	bis zu mehreren Mio. Gatteräquivalenten

9. Kapitel: Automaten

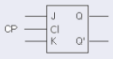


Prinzipieller Aufbau eines synchronen Schaltwerks

- Definition: Elementare Maschine zur Verarbeitung diskreter Eingangsgrößen und Erzeugung diskreter Ausgangsgrößen. Interne Speicher definieren Zustände.
- Synchrone Schaltwerke: Taktsynchroner Übergang von stabilen Zustand in stabilen Folgezustand
- Asynchrone Schaltwerke: Asynchroner Zustandswechsel mit Wechsel der Eingangsgrößen

Elementares Schaltwerk: JK-FF

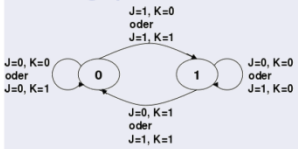
Schaltsymbol:



Zustandsübergangstabelle:

J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

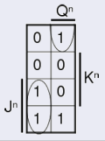
Zustandsgraph:



Zustandsübergangsgleichung:

$$Q_{n+1} = \overline{K_n}Q_n \vee J_n\overline{Q_n} = \overline{K_n}Q_n \vee J_n\overline{Q_n}$$

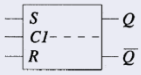
KV-Diagramm:



Hinweis: Zustandsübergang bei Anliegen der genannten Bedingungen zum Zeitpunkt des Wirksamwerdens des Takts

Elementares Schaltwerk: RS-FF

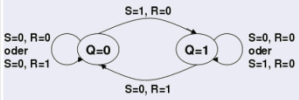
Schaltsymbol:



Zustandsübergangstabelle:

S_n	R_n	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1

Zustandsgraph:



Hinweis: Zustandsübergang bei Anliegen der genannten Bedingungen zum Zeitpunkt des Wirksamwerdens des Takts

Zustandsübergangsgleichung:

$$Q_{n+1} = S_n \vee \overline{R_n}Q_n$$

Zusatzbedingung:

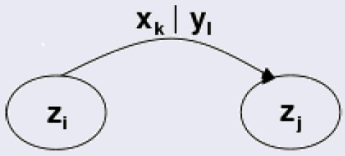
$$S_nR_n = 0$$

- Übliche **formale Beschreibung** eines Automaten: M = (X, Z, Y, z_0, f g)
  - Eingabealphabet X = $x_{1, \dots, } x_n$
  - Ausgabealphabet Y = $y_{1, \dots, } y_m$
  - Zustandsmenge Z = $z_1, \dots, z_l$
  - Anfangszustand $z_0 \in Z$
  - Übergangsfunktion $g(x_i, z_j) \rightarrow z_k$
  - Ausgangsfunktion $f(x_i, j_j) \rightarrow y_k$

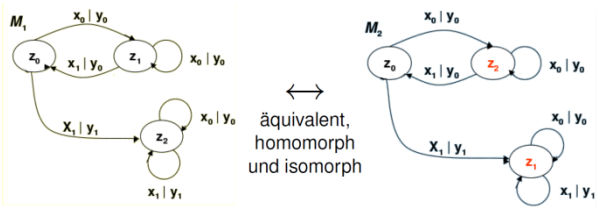
Zustandübergangstabelle

$z_0, \dots, z_l$	$x_0, \dots, x_n$	$z_0^+, \dots, z_l^+$	$y_0, \dots, y_m$

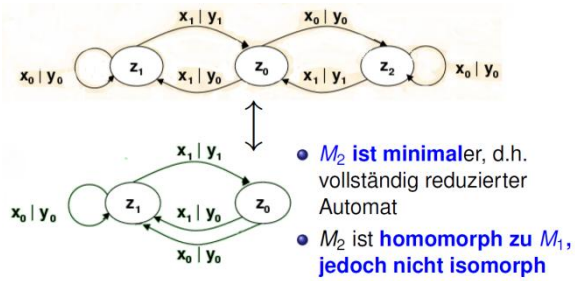
Zustandsgraph



- **Zustandsgraph: Eigenschaften**
  - Bei n Speicherglieder höchstens 2^n Knoten
  - Bei m Eingangsvariablen gehen von jedem Knoten maximal 2^m Kanten aus.
- **Endlichkeit und Vollständigkeit**
  - Ein Automat M heißt endlich, wenn Eingabe- und Ausgabealphabet endlich sind.
  - Ein Automat M heißt deterministisch, wenn gilt für alle $x \in X, z \in Z : | (g(x, z) | \leq 1$ und wenn ein Folgezustand für x und z existiert.
  - Ein Automat M heißt vollständig, wenn gilt für alle $x \in X, z \in Z : | (g(x, z) | \leq 1$ und wenn immer mindestens ein Folgezustand existiert.
- **Äquivalenz**
  - 1-äquivalent, falls beide Zustände sich am Ausgang identisch verhalten
  - 2-äquivalent, falls zudem auch die beiden Folgezustände sich am Ausgang identisch verhalten
  - n-äquivalent (äquivalent), falls sich für jede Eingangsfolge von z_1 und z_2 dieselbe Ausgangsfolge einstellt
  - Zwei Automaten M und M‘ heißen äquivalent, falls zu jedem Zustand des Automaten M ein äquivalenter Zustand des Automaten M‘ existiert und umgekehrt.
- Ein Automat M heißt **minimal**, falls die Anzahl der Zustände minimal ist



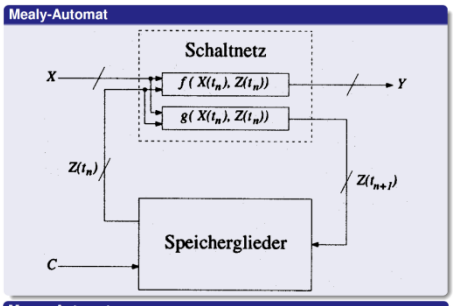
Isomorphismus: Veranschaulichung

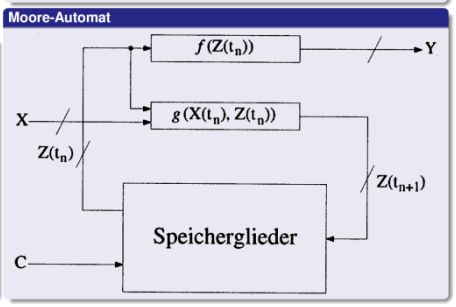


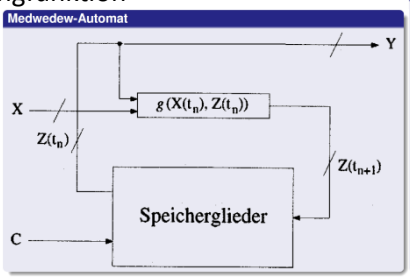
- M_2 **ist minimaler**, d.h. vollständig reduzierter Automat
- M_2 ist **homomorph zu M_1, jedoch nicht isomorph**

Homomorphismus: Veranschaulichung

- Automatentypen
  - Mealy-Automat: y = f(x,z)
    Ausgangsfunktion ist kombinatorische Verknüpfung aus X und Z
  - Moore-Automat: y = f(z)
    Y hängt nur Z ab, nicht vom X
  - Medwedew-Automat: y = id(z) = z
    Y ergibt sich direkt aus Z damit hat damit keine Ausgangfunktion

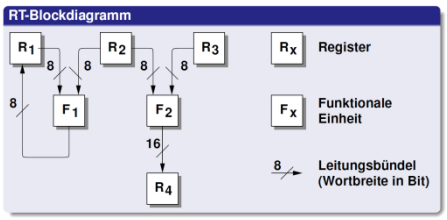


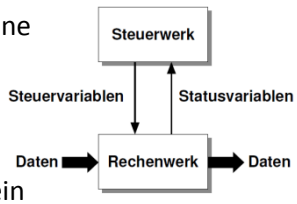




10. Kapitel: RT-Ebene

- Entwurf komplexer System mit Beschreibung auf Gatterebene schwierig
- Beschreibung auf **Register-Transfer-Ebene**
  - Aufteilung in speichernde und verarbeitende Einheiten
  - Register zur Speicherung
  - Funktionale Einheiten (Zähler, Addierer, Mux, …)
- Zusammenfassen der Datenwörter zu Feldern
- Aufteilung des Entwurfs in Rechenwerk/Operationswerk und Steuerwerk.
- **Typen von Bauelementen**
  - *Register* als Datenwortspeicher
  - *Multiplexer und Demultiplexer* zur Schaltung von Verarbeitungswegen
  - *Verarbeitungsschaltnetze*, d.h. das Operationswerk, zur Realisierung der Funktionen
  - *Steuerwerk* zur Erzeugung der Steuersignale für obige Bausteine
- Entwurf des Steuerwerks unabhängig vom Entwurf des Rechenwerks
- **Umsetzung in HW-Beschreibungssprache**
  - Variable R_i entspricht Speicherbaustein
  - Verknüpfung F_i von Variablen entspricht Verarbeitungsbaustein
  - Register-Transfer-Operation: $R_i = F_i(R_j, R_k)$
- Unterscheidung zwischen Bit- und Wortzugriff: Z<i> selektiert Bit i, Z[i] selektiert Wort i
- Beispiele: Register A, B mit je 8 Bit Breite: register A<0:7>, B<0:7>;
  Speicher C mit 4k Worten zu je 12 Bit: memory C[0:4095]<0:11>;
  Konkatenation zweier Variablen zu Verbundsregister: register AB=A.B;





- **Operationen auf Variablen**
  - Zuweisung von Werten: A:=B;
  - Steuerung der Nebenläufigkeit:
    - Nebenläufige Zuweisung, Vertauschung: A:=B, B:=A;
    - Sequentielle Zuweisung, d.h. A=B: A:=B; B:=B;
  - Verknüpfung von Variablen: B:= A+B;
  - Nutzung eines Teils der Variablen: AB<0:11>:=C[127]; B:=A+C[127]<0:7>;
  - If COND then A:=B else A:=C;
- FFs, Register und Speicher werden nur abstrakt dargestellt, keine Details über Bits/Worte.
- **Verbindungsarten**
  - Punkt-zu-Punkt-Verbindungen (exklusiver Kommunikationspfad)
  - Busse (verbinden Geräte miteinander/Gemeinsam genutzter Kommunikationspfad)
- **Arte von Bussen**
  - Unterscheidung nach **Datenflußrichtung**
    - *Unidirektionaler Bus*, d.h. Information fließt in nur eine Richtung; Teilnehmer ist Quelle oder Senke, nicht beides.
    - *Bidirektionaler Bus*, d.h. Information fließt in beide Richtungen; Teilnehmer ist Quelle und/oder Senke
  - Unterscheidung nach **Teilnehmeradressierung**
    - *Broadcasting Bus*, d.h. alle Empfänger (Senken) empfangen die vom Sender (Quelle) auf den Bus gegebenen Informationen
    - *Selective Addressing Bus*, d.h. Ansprechen ausgewählter Empfänger
- **Vermeidung von Buskollisionen**
  - Buszugriff nur exklusiv, sonst Kollision
    → Steuersignale zur Ankopplung von Bausteinen
  - An- und Abkopplung mit Tri-State-Puffern
- **Schreib- und Lesespeicher**
  - Beliebige Adressierung von Speicherzellen in Schreib-/Lesespeicher erfordert wahlfreien Zugriff → Random-access Memory (RAM)
  - Schreibender und lesender Zugriff erforderlich (zeitgleiches Schreiben und Lesen über einzelnen Bus nicht möglich → gemeinsames Schreib/Lesesignal zur Steuerung)
- **Rechenwerk**
  - Wesentliche Verarbeitungseinheit in Rechnern
  - Datenwerk mit über ALU verbundenen Register-/Speicherelementen
- **Arten von Steuerwerken**
  - *Festverdrahtete Steuerwerke*: Unveränderliche Realisierung mittels fester Bauelemente, z.B. Gatter und Multiplexer (hard-wired)
  - *Mikroprogrammierte Steuerwerke*: Ablegen von Zuständen in Speicher, Speicheradresse definiert Zustand, Speicherinhalt definiert Steuerbits, Sequentieller Abruf von Adressen → Zustandssequenz (microprogrammed)
- **Mikroprogramm**: Folge von Mikrobefehlen, die erforderlich ist, um einen Maschinenbefehl auszuführen oder einen Systemzustandswechsel durchzuführen. Das Mikroprogramm wird auf einer Wirtsmaschine abgearbeitet, welche die benötigte Basishardware bereitstellt.
- **Horizontale Mikroprogrammierung**

  - Implizite Zuordnung zwischen Stellen des Mikroinstruktionswortes und der Rechnerhardware mit parallelem Anstoßen der so kodierten Mikrooperationen
  - Für jedes Steuersignal wird ein Bit im Befehlswort vorgesehen
  - Vorteil: Steuersignale können unabhängig voneinander gesetzt werden (Parallelität)
  - Nachteil: großer Mikroprogrammspeicher erfordert,
    oft nicht alle Kombinationen der Steuersignale sinnvoll oder erlaubt
- **Vertikale Mikroprogrammierung**
  - Zuordnung zwischen Stellen des Mikroprogrammwortes und den Mikrooperationen durch einen verschlüsselten Mikrooperationscode
  - Nur ein Kontrollfeld mit Id(m) Bit wird in der Mikroinstruktion gespeichert
  - Wert des Kontrollfelds ist Binärcodierung des zu setzenden Steuersignals
  - Vorteil: Kleiner Mikroprogrammspeicher
  - Nachteil: Es kann immer nur ein Steuersignal gesetzt werden (keine Parallelität)
- Quasi-horizontale Mikroprogrammierung
  - Wie horizontale, jedoch mit Zerlegung der Menge der Mikrooperationen in Mengen wechselseitig unvereinbarer Mikrooperationen mit Kodierung der einzelnen Teilmengen
  - Zusammenfassung der Steuersignale, die nicht gleichzeitig aktiv sein müssen/dürfen und Codierung der Gruppen
  - Anzahl der Felder entspricht Parallelitätsgrad
  - Vorteil: Kompakteres Mikroprogrammwort, idealerweise ohne Verschwendung von Parallelisierungsmöglichkeiten
  - Nachteil: Dekoderstufen notwendig (Durchlaufzeit, Hardwareaufwand)

Horizontale Mikroprogrammierung

Vertikale Mikroprogrammierung

Quasi-horizontale Mikroprogrammierung

- Aufteilung in Teilsteuerwerke: Zerlegung eines komplexen Steuerwerks in mehrere einfachere Steuerwerke → leichterer Entwurf der Teilsteuerwerke
- Mikroprogrammierbares System
  - Grobgranulare Struktur mit komplexen Funktionsblöcken
  - Datentransport- und Ablaufkontrolle mittels Mikroprogramm
  - Ausnutzung von Parallelismus möglich bei (quasi-)horizontaler Mikroprogrammierung
  - Übergeordnete sequentielle Ablaufsteuerung über Makroprogramm
  - Rekonfiguration rein auf Softwarebene

11. Kapitel: RT-Entwurf

- Ablauf eines MPs für die Addition

```
do
if ADDIEREN,
then while C<>0
do A:=A xor C, C:= C and A;
   C:= C <<1;
end;
```

- Ablauf einer Multiplikation

Befehlssatz einer einfachen CPU

Makroinstr.	Opcode	Beschreibung	Bedeutung
LDA <addr>	0000	load A from <addr>	A:=RAM[<addr>]
LDB <addr>	0001	load B from <addr>	B:=RAM[<addr>]
STA <addr>	0010	store A to <addr>	RAM[<addr>]:=A
STB <addr>	0011	store B to <addr>	RAM[<addr>]:=B
ADD	0100	add	A:=A+B
SUB	0101	sub	A:=A-B
TAB	0110	transfer A to B	B:=A
TBA	0111	transfer B to A	A:=B
JMP <addr>	1000	jump to <addr>	PC:=<addr>
JN <addr>	1001	jump to <addr> if neg.	PC:=<addr> falls A<0
NOP	101-	no operation	keine Operation
HLT	11--	halt	Anhalten