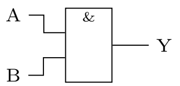
Digitaltechnik

Andrej Scheuer
ascheuer@student.ethz.ch
6. Dezember 2020

Gates

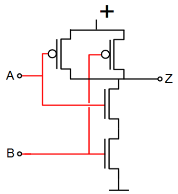
AND

$Y = A \wedge B \quad Y = A \cdot B$

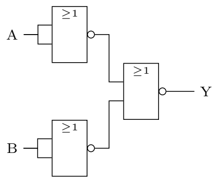


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

NAND

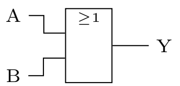


AND aus NOR



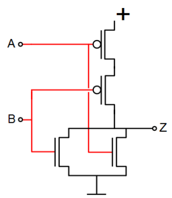
OR

$Y = A \vee B \quad Y = A + B$

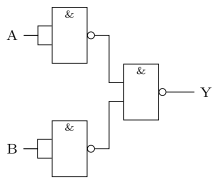


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

NOR

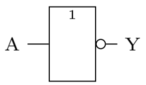


OR aus NAND

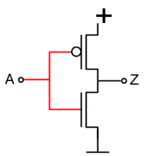


NOT

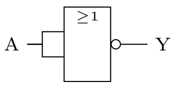
$Y = \overline{A}$



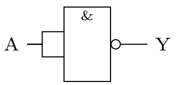
A	Y
0	1
1	0



NOT aus NOR



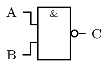
NOT aus NAND



Weitere Gates

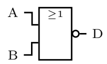
NAND

$C = \overline{A \wedge B}$



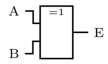
NOR

$D = \overline{A \vee B}$



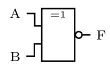
XOR

$E = A \oplus B$



XNOR

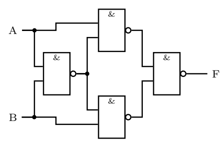
$F = \overline{A \oplus B}$



A	B	C (NAND)	D (NOR)	E (XOR)	F (XNOR)
0	0	1	1	0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1

$$\text{XOR} = (A \wedge \overline{B}) \vee (\overline{A} \wedge B)$$
$$\text{XNOR} = (A \wedge B) \vee (\overline{A \wedge B})$$

XOR aus NAND



XOR aus NOR: Gleiches Schema wie NAND + 1 Inverter

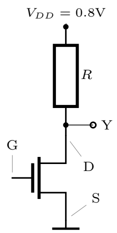
XNOR aus NAND: Gleiches Schema wie *XOR aus NOR*

XNOR aus NOR: Gleiches Schema wie *XOR aus NAND*

Es versteht sich natürlich, dass wenn von „Gleichem Schema wie…" gesprochen wird, die Gates trotzdem getauscht werden müssen.

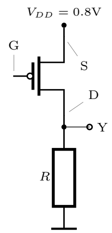
CMOS

NMOS



G	Schalter	Y
0	offen	1
1	zu	0

PMOS



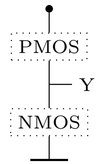
G	Schalter	Y
0	zu	1
1	offen	0

Konstruktion von CMOS-Gates

Regeln für CMOS-Schaltungen

1. CMOS-Gates bestehen aus gleich vielen NMOS und PMOS.
2. m Eingänge: m NMOS und m PMOS.
3. NMOS in Serie → PMOS parallel
4. NMOS parallel → PMOS Serie

Allg. Aufbau CMOS

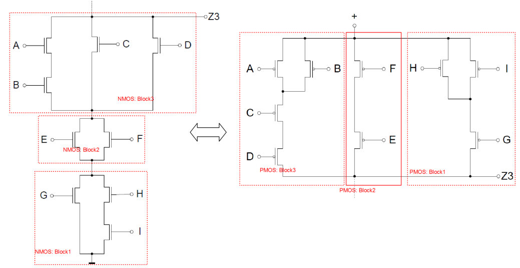


Pull-up: PMOS
Pull-down: NMOS

Pfade sind komplementär (**Serie** ⇔ **Parallel**)

Umwandlung Pull-up zu Pull-down

1. Teilbereiche (Blöcke) identifizieren.
2. Schritt 1 wiederholen, bis nur noch einzelne Transistoren vorkommen.
3. Falls Pull-down:
   - Von GND aus mit äusserstem Block beginnen.
   - PMOS → NMOS
4. Falls Pull-up:
   - Von V_{DD} aus mit äusserstem Block beginnen.
   - NMOS → PMOS.



Funktionsgleichung

parallel: $\vee$	Pull-Up: $y = 1$	alle I : 0 → I invert.
Serie: $\wedge$	Pull-Down: $y = 0$	alle I : 1 → Gl. invert.

Boolsche Algebra

Grundregeln

Kommutativität

$$A \wedge B = B \wedge A$$
$$A \vee B = B \vee A$$

Assoziativität

$$A \wedge (B \wedge C) = (A \wedge B) \wedge C$$
$$A \vee (B \vee C) = (A \vee B) \vee C$$

Distributivität

$$(A \wedge B) \vee (A \wedge C) = A \wedge (B \vee C)$$
$$(A \vee B) \wedge (A \vee C) = A \vee (B \wedge C)$$

Nicht	$\overline{\overline{A}} = A$	
Null-Th.	$A \vee 0 = A$	$A \wedge 0 = 0$
Eins-Th.	$A \vee 1 = 1$	$A \wedge 1 = A$
Idempotenz	$A \vee A = A$	$A \wedge A = A$
V. Komp.	$A \vee \overline{A} = 1$	$A \wedge \overline{A} = 0$
Adsorp.	$A \vee (\overline{A} \wedge B) = A \vee B$	
	$A \wedge (\overline{A} \vee B) = A \wedge B$	
Adsorp.	$A \vee (A \wedge B) = A$	
	$A \wedge (A \vee B) = A$	
Nachbar.G.	$(A \wedge B) \vee (\overline{A} \wedge B) = B$	
	$(A \vee B) \wedge (\overline{A} \vee B) = B$	

De Morgan

1. Regel $\overline{A \wedge B} = \overline{A} \vee \overline{B}$
2. Regel $\overline{A \vee B} = \overline{A} \wedge \overline{B}$

Regeln gelten auch für n verknüpfte Terme.

Normalformen

Minterm	**Maxterm**
AND-Ausdruck	OR-Ausdruck
Output: 1	Output: 0
n Schaltvar. → 2^n mögl. Minterme.	n Schaltvar. → 2^n mögl. Maxterme.
nicht-invertierte Var: 1	nicht-invertierte Var: 0
invertierte Var: 0	invertierte Var: 0

Disjunktive Normalform

1. Identifiziere WT-Zeilen mit Output 1
2. **Minterme** für diese Zeilen aufstellen
3. Minterme mit **OR** verknüpfen

Konjunktive Normalform

1. Identifiziere WT-Zeilen mit Output 0
2. **Maxterme** für diese Zeilen aufstellen
3. Maxterme mit **AND** verknüpfen

A	B	Y	Minterm	Maxterm
0	0	1	$\overline{A} \wedge \overline{B}$	
0	1	0		$A \vee \overline{B}$
1	0	0		$\overline{A} \vee B$
1	1	1	$A \wedge B$	

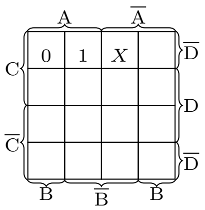
DNF	$Y = (\overline{A} \wedge \overline{B}) \vee (A \wedge B)$	1 Mint. erf. → 1
KNF	$Y = (A \vee \overline{B}) \wedge (\overline{A} \vee B)$	1 Maxt. erf. → 0

NAND/NOR Schaltungen

Schaltung nur aus:

- NAND: DNF → 2× Negieren → 1× De Morgan
- NOR: KNF → 2× Negieren → 1× De Morgan oder: auf jeden Term der DNF De Morgan anwenden.

Karnaugh Diagramme (KVD)

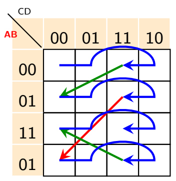


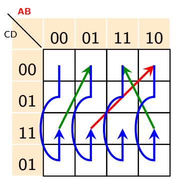
AB \ CD	00	01	11	10
00	0	1	X	
01				
11				
01				

Hat das Karnaugh Diagramm 5 Dimensionen, wird die 5te Dimension auf zwei Tabellen aufgeteilt.

Don't-Care-Zustände $X \in \{0,1\}$ Redundante, überflüssige oder unmögliche Kombinationen der Eingangsvariablen werden mit einem X markiert.

Schema zum Ausfüllen





Päckchen

- Päckchen immer rechteckig (Ausnahme: über Ecken).
- Umfassen möglichst grosse Zweierpotenz.
- Dürfen über Ecken und Grenzen hinausgehen und sich überlappen.

DNF

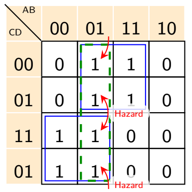
1. KVD ausfüllen.
2. Päckchen mit **1** uo X.
3. Vereinfachte Minterme aufstellen.
4. Minterme mit OR verbinden.

KNF

1. KVD ausfüllen.
2. Päckchen mit **0** uo X.
3. Vereinfachte Maxterme aufstellen.
4. Maxterme mit AND verbinden.

Hazard

Kurzzeitige, unerwünschte Änderung der Signalwerte, die durch Zeitverzögerung der Gatter entstehen.



Statische Hazards Stellen im KVD, an denen sich Päckchen orthogonal berühren, aber nicht überlappen.
Lösung Berührende Päckchen mit zusätzlichen (möglichst grossen) Päckchen verbinden.

Zahlensysteme

D	zu berechnende positive Zahl
R	Basis/Radix von D
b_i	Koeffizient

$$D = \sum_{-\infty}^{\infty} b_i \cdot R^i$$

Darstellung D in Basis R: $\ldots b_2 b_1 b_0 . b_{-1} b_{-2} \ldots_R$

Dezimal	10	$b_i \in \{0,1,\ldots,9\}$
Dual/Binär	2	$b_i \in \{0,1\}$
Oktal	8	$b_i \in \{0,1,\ldots,7\}$
Hexa	16	$b_i \in \{0,1,\ldots,9,A,B,C,D,E,F\}$

Umwandlung Zahlensysteme

1. Ganzzahlige Division mit R: $D/R = Q_0 + r_0$.
2. $$Q_i/R = Q_{i+1} + r_{i+1}$$
   bis $Q_i = 0$.
3. Erste Operation gibt MSB, letze Operation gibt LSB (aka. unten nach oben lesen.)

Für $1 > D \geq 0$

$$D \cdot R = P_0 \quad K_{-1} = \text{floor}\,(P_0) \quad a_{-1} = P_0 - K_{-1}$$
$$a_{-1} \cdot R = P_{-1} \ldots$$

K_i: Koeffizienten für Zahlensystem. Erste Operation gibt **MSB**, letze Operation gibt **LSB** (aka von oben nach unten lesen).

Dezimal zu Binär

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

2^{-1}	2^{-2}	2^{-3}	2^{-4}
0.5	0.25	0.125	0.0625

Binär zu Hex

0000	0	0100	4	1000	8	1100	C
0001	1	0101	5	1001	9	1101	D
0010	2	0110	6	1010	A	1110	E
0011	3	0111	7	1011	B	1111	F

Zweierkomplement

Sign Bit
0: positiv 1: negativ

Konstruktion

1. Zahl $|Z|$ in Binär B umwandeln.
2. B bitweise invertieren
3. 1 zu LSB addieren (! Übertrag)
4. Sign Bit hinzufügen (zuvorderst).

Ist die Blocklänge länger als Zahl, vorangehende 0(-en) miteinbeziehen.

2er Komplement zu Dezimal

$$D_{(10)} = -b_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} b_i \cdot 2^i$$

Wertebereich 2er-Komp. $\left[-2^{n-1}, 2^{n-1} - 1\right]$

mQn

$$D_{(10)} = -b_m \cdot 2^m + \sum_{i=0}^{m-1} b_i \cdot 2^i + \sum_{i=1}^{n} b_i \cdot 2^{-i}$$

m: Vorkommabits, n: Nachkommabits
Sign-Bit muss nur einmal vor dem m codiert werden.

Binäre Rechenoperationen

Addition

Bitweise Addition der Binärzahlen. Leere Slots werden mit 0 aufgefüllt.

Subtraktion

Addition via 2er Komp. Übertrag von MSB ignorieren.

Multiplikation

- Bitweise Multiplikation des Multiplikanden a mit b_i des Multiplikator.
- Sukzessive Multiplikation werden um ein Bit (0) nach links verschoben.
- Anzahl Nachkommabits ergibt sich aus der Summe der Anzahl Nachk.bits der Operatoren.

$$\begin{array}{r} b_0 \cdot a \\ +b_1 \cdot a\ 0 \\ +b_2 \cdot a\ 0\ 0 \\ +b_3 \cdot a\ 0\ 0\ 0 \\ \hline = \text{Sum} \end{array}$$

Division

1. Identifiziere Teil des Divident $>$ Divisor (Unterblock). Für jede Stelle, sodass Divident $<$ Divisor, 0 in Quotient.
2. Unterblock $-$ Divisor, 1 an Quotient anhängen, Rest behalten.
3. An das Resultat der Subtraktion Bits des Dividenten anhängen. Wiederholen bis Subtraktion 0 ergibt.

Parity-Bits

Hilft Bit-Fehler zu finden.
Bitsequenz wird in 4 Bits unterteilt. Pro Nibble wird ein Parity-Bit angefügt. Nach 4 Blöcken folgt ein Prüfwort.

Parity-Bit	Anz. 1	PB	Nibble + PB
Even P_E	ungerade	1	gerade
	gerade	0	
Odd P_O	ungerade	0	ungerade
	gerade	1	

01010 11011 10111 00101 00011

Korrekt P_E

0	1	0	1	0
1	1	0	1	1
1	0	1	1	1
0	0	1	0	1
0	0	0	1	1

Fehler P_E

0	1	0	1	0
1	1	1	1	1
1	0	1	1	1
0	0	1	0	1
0	0	0	1	1

Latches und FlipFlops

Kombinatorische Schaltung Output hängt von Inputs und Verknüpfungen ab.

Sequentielle Schaltung Enthält Rückkopplungen, Outputs hängen von vorherigen Werten ab.

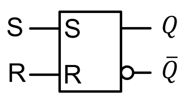
Latch
(Takt)zustandgesteurte Schaltung $\rightarrow$ Änderungen am Eingang können während der **ganzen aktiven Taktphase** den Output beeinflussen.

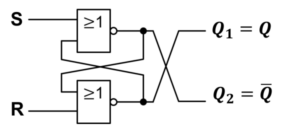
FlipFlops
Taktflankengesteuerte Schaltung $\rightarrow$ Input zum Zeitpunkt der Taktwechsels wird wirksam.

Latches

Alle taktzustandgesteurte Schaltungen sind gegenüber Störimpulsen empfindlich, da bei T = 1 jede Änderung übernommen wird.

SR-Latch



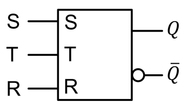


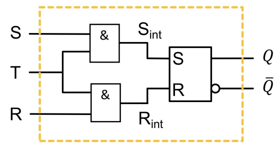
S	Set	$\rightarrow$	setzt Q auf 1
R	Reset	$\rightarrow$	setzt Q auf 0

$$Q_{n+1} = S \vee \left(Q_n \wedge \overline{R}\right)$$

Fall	S	R	Q_{n+1}	
1	0	0	Q_n	speichern
2	0	1	0	zurücksetzten
3	1	0	1	setzen
4	1	1	-	unzulässig

SRT-Latch

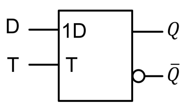


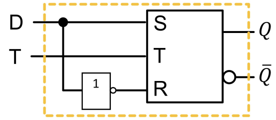


T		S_{int}	R_{int}		
0	$\rightarrow$	0	0	$\rightarrow$	Datenspeicherung
1	$\rightarrow$	S	R	$\rightarrow$	Normales SR-Latch

Änderungen werden nur übernommen, wenn T/CLK aktiv ist.

D-Latch

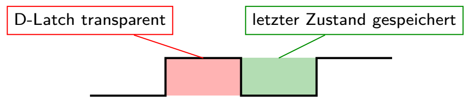




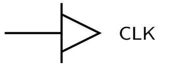
Bauelement, das Daten für die Periodendauer eines Taktes speichern kann.

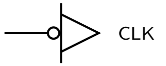
$$Q_{n+1} = \left(Q_n \wedge \overline{\text{T}}\right) \vee (\text{D} \wedge \text{T})$$

T	Q_{n+1}		
0	Q_n	$\rightarrow$	alter Ausgang gespeichert
1	D	$\rightarrow$	Input übernommen



FlipFlops





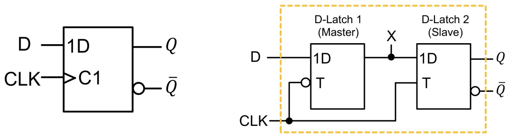
Input beim Übergang von **0** → **1** von CLK wirksam.

Input beim Übergang von **1** → **0** von CLK wirksam.

Positive/steigende Taktflanke

Negative/fallende Taktflanke

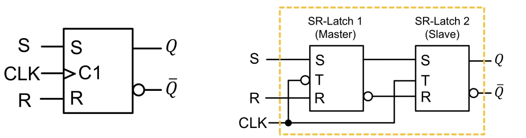
D-FlipFlop



$Q_{n+1} = \text{D}$ wenn CLK $0 \to 1$

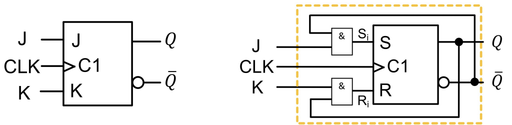
Master	low-active	CLK $= 0$
Slave	high-active	CLK $= 1$

SR-FlipFlop



$Q_{n+1} = \text{S} \vee \left(\overline{\text{R}} \wedge Q_n\right)$ wenn CLK $0 \to 1$

JK-FlipFlop



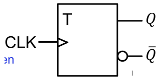
$Q_{n+1} = \left(J \wedge \overline{Q_n}\right) \vee \left(\overline{K} \wedge Q_n\right)$ wenn CLK $0 \to 1$

Fall	J	K	Q_{1n+1}	Q_{2n+1}	
1	0	0	Q_{1n}	Q_{2n}	speichern
2	0	1	0	1	zurücksetzten
3	1	0	1	0	setzen
4	1	1	$\overline{Q_{1n}}$	$\overline{Q_{2n}}$	wechseln

Bei J = K = 1 wechselt Output. (toggel)

T-FlipFlop

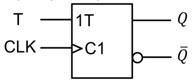
V1 Ausgang wechselt bei jeder aktiven Taktflanke.



$Q_{n+1} = \overline{Q_n}$

wenn CLK $0 \to 1$

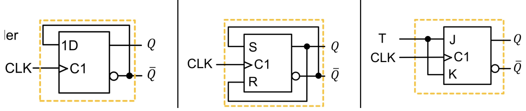
V2 Ausgang wechselt bei aktiver Taktflanke nur wenn $T = 1$.



$Q_{n+1} = \overline{Q_n}$

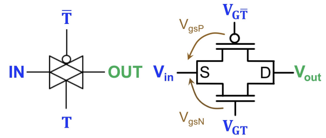
wenn CLK $0 \to 1 \wedge T = 1$

BS V1 | BS V1 | BS V2



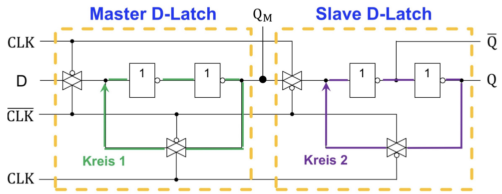
D-FlipFlop in CMOS-Technik

Transmission Gates

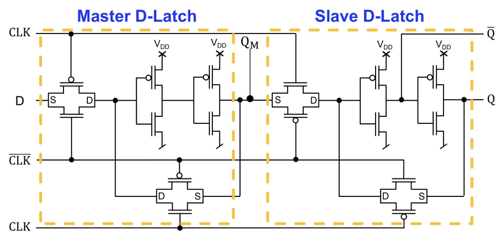


IN	T	Widerstand	OUT
0	0	hochohm.	-
0	1	niederohm.	0
1	0	hochohm.	-
1	1	niederohm.	1

TG sperrt wenn Widerstand hochohmig ist. (T $= 0$)



CLK	
0	Input ins erste Latch übertragen
1	Latch verriegelt, Wert im Kreis gefangen



D-FlipFlop ⇔ JK-FlipFlop

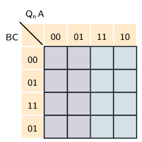
1. JK-FF kann immer durch D-FF ersetzt werden.

   D-FF: $D_n = \left(J \wedge \overline{Q_n}\right) \vee \left(\overline{K} \wedge Q_n\right)$:JK-FF

2. Ein D-FF kann nur durch JK-FF ersetzt werden wenn:
   a) Schaltung eine Rückkopplung enthält.
   b) Input D als $\left(F_1 \wedge \overline{Q_n}\right) \vee (F_2 \wedge Q_n)$ geschrieben werden kann.

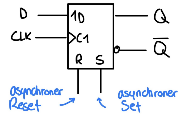
Gleichung für D-FF → JK-FF

1. Wahrheitstabelle mit Einängen und Rückkopplung.
2. Wahrheitstabelle in Q_n und $\overline{Q_n}$.
3. Separat Päckchen in Q_n und $\overline{Q_n}$ machen.
4. Päckchen mit OR verbinden. Ggf. Q_n und $\overline{Q_n}$ ausklammern.



Asynchroner Set/Reset Input

Können gespeicherte Zustände asynchron zu CLK überschreiben.



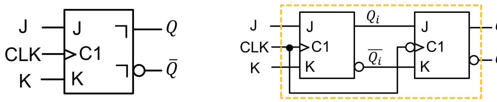
Verzögerungszeiten

t_s	Setup-Zeit	Solange muss Signal vor aktiver Taktflanke stabil anliegen.
t_h	Hold-Zeit	Solange muss Signal nach aktiver Taktflanke stabil anliegen.
t_{pd}	Verzögerungszeit	Durchlaufzeit

$$T_{\text{min}} \geq t_{\text{pd1}} + t_{\text{pd,ks}} + t_{\text{s2}} \qquad f_{\text{max}} = \frac{1}{T_{\text{min}}}$$

t_h kann bei der Berechnung von f_{max} vernachlässigt werden. Es wird der längste PFad zwischen zwei FlipFlops betrachtet.

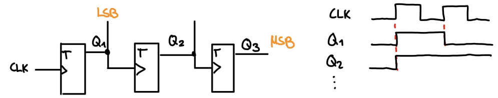
Zwischenspeicher-FF



FlipFlop, dass Input bei steigender Taktflanke übernimmt und bei der nächsten fallenden Taktflanke ausgibt. (oder umgekehrte Flanken)

⌝ Ausgabe bei fallender Flanke
⌟ Ausgabe bei steigender Flanke

Frequenzteiler und Zähler



Kaskadieren von T-Flipflops führt zu einer Frequenzreduktion von CLK um Faktor 2.
Kann als Bitzähler verwendet werden (ohne CLK). MSB ist längste Frequenz. $n_{T,ff} \to 0 \dots (2^n - 1)$

Automaten

Ein System, das auf seine Eingänge reagiert und einen Ausgang produziert, der vom Eingangssignal und momentanen Zustand abhängt.
Bei synchronen Automaten besitzen alle Speicherelemente (FlipFlops) den gleichen Takteingang.

Formale Beschreibung

$X = (x_1, \dots, x_e)$	Eingangsalphabet mit **e** Eingängen
$Y = (y_1, \dots, y_b)$	Ausgangsalphabet mit **b** Ausgängen
$Z = (z_1, \dots, z_m)$	Zustandsmenge mit **m** internen Zuständen
$Z_0 \in Z$	Anfangszustand
$f_{c1} : (X_n, Z_n) \to Z_{n+1}$	Übergangsfunktion
$f_{c2} : (X_n, Z_n) \to Y_n$	Ausgangsfunktion

Automatentypen

Mealy	Ausgänge von inneren Zuständen und Eingängen abhängig $Y_n = f_{c2}(X_n, Z_n)$
Moore	Ausgänge nur von inneren Zuständen abhängig (keine Verbindung zwischen Input und Output) $Y_n = f_{c2}(Z_n)$
Medwedjew	Ausgänge entsprechen inneren Zuständen $Y_n = Z_n$

Zustandsfolgetabelle

Auflistung aller möglichen Kombinationen der aktuellen inneren Zuständen sowie den Eingängen mit den dazugehörigen Folgezuständen und Ausgängen.

$x_1 \dots x_e$	$z_{1n} \dots z_{mn}$	$z_{1(n+1)} \dots z_{m(n+1)}$	$y_1 \dots y_b$

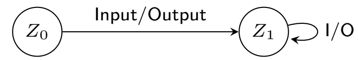
$e + 2m + b$ Spalten

2^{e+m} Zeilen

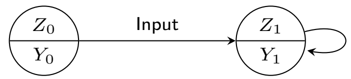
Zustandsdiagram

Knoten	interne Zustände
Kanten	Übergänge zwischen Zuständen

Mealy-Automat



Moore-Automat



Wichtig Von jedem Knoten aus muss es für jeden Eingang eine Kante geben, diese können aber zusammengefasst werden.

Entwurf eines Automaten

1. Auftrag lesen und analysieren → Automatentyp bestimmen.
2. Zustandsmenge bestimmen → Anzahl erforderlich D-FlipFlops $\lceil \log_2(\text{Anzahl Zustände}) \rceil$.
3. Eingangs- und Ausgangsvariablen definieren, Kodierung.
4. Darstellung der Zustandsfolge in einem Zustandsdiagramm.
5. Zustandsfolgetabelle aufstellen.
6. Minimierte Ausgangs- und Übergangsfunktion bestimmen mit KV-Diagrammen bestimmen.
7. Unbenutzte Zustände überprüfen.
8. Schaltplan anhand Schaltfunktion konstruieren.

Umwandlung Mealy ⇔ Moore

Moore → Mealy

1. Ausgänge von Folgezuständen auf Kanten schreiben.
2. Ausgänge bei Zuständen entfernen.

Mealy → Moore

1. Ausgänge in Knoten schreiben, an denen Kante endet.
2. Knoten mit mehr als einem Ausgang multiplizieren → neu kodieren.
3. Eingehende Kanten entsprechend der Ausgänge auf neue Knoten umhängen.
4. Ausgehende Kanten für alle neue Knoten kopieren.

Diese Umwandlung ist immer möglich, aber meistens werden mehr Zustände benötigt.
Wichtig: Das Zeitverhalten der Ausgänge verändert sich bei der Umwandlung.

Mealy	Eingangsveränderungen beeinflussen den Ausgang sofort.
Moore	Eingangsveränderungen haben erst bei Taktflanke Einfluss (weniger Störungsanfällig)

Diverses

Physikalische Zuordnung logischer Zustände

0	Low $0\,\text{V}$	Ground
1	High $0.8\,\text{V}$	VDD

Toleranzen:
- GND: $0\,\text{V} \ldots 0.15\,\text{V}$
- VDD $0.7\,\text{V} \ldots 0.9\,\text{V}$

Schaltelemente

Multiplexer

Sendet eines von 2^n Eingangssignalen an den Ausgang. Hat n Auswahlbits.

Demultiplexer

Sendet 1 Eingangssignal an einen von 2^n Ausgänge. n Auswahlbits.

Halbaddierer

Addiert 2 Binärzahlen A und B. Produziert Summe und Carry-Out.

$$\text{SUM} = A \oplus B \qquad \text{CO} = A \wedge B$$

Volladdierer

Nimmt einen zusätzlichen Input CI entgegen.

$$\text{SUM} = (A \oplus B) \oplus CI \qquad \text{CO} = (A \wedge B) \vee (S_{AB} \wedge CI)$$

Serienaddierer

Addition einer Stelle pro Taktschritt.

Paralleladdierer (Normalform)

Addition aller Stellen pro Taktschritt.

Vorteile
- Maximal 3 Grundgatter zwischen Input und Output.
- Laufzeit ist unabhängig von Stellenzahl der Summanden.

Nachteile Bei Addition von n-stelligen Summanden müssen $\sim n \cdot 2^{2n-1}$ Min-/Maxterme verknüpft werden.

→ Schnell aber Schaltungsaufwendig

Ripple-Carry Addierer (Paralleladdierer)

Vorteile
- Durch Kaskadierung einfach skalierbar.
- Schaltungsaufwand linear zur Stellenzahl.

Nachteile
- SUM und CO für die i-te Stelle können erst nach der Berechnung der $(i-1)$-ten Stelle gebildet werden.
- Addierzeit linear zu Stellenzahl

Langsamer als Normalformaddierer aber einfacher zu realisieren.

Carry-Look-Ahead Addierer (Paralleladdierer)

Kombination der Vorteile des Normalform- und Ripple-Carry-Addierer → schnelle Schaltung mit begrenztem Aufwand.

Praktische Realisierung Addierer werden kaskadiert, Berechnung der Überträge erfolgt parallel zur Summenbildung.
Berechnungsaufwand ist linear zur Stellenzahl, Laufzeit bleibt konstant.

Booth-Algorithmus

Dient der Multiplikation von Binärzahlen (A & B).
Berechnung über Zwischenprodukte P_i.
Division durch 2 bedeutet: Verschiebung des Kommas nach links (shift), mit Vorzeichenverdoppelung falls nötig.

a_i	a_{i-1}	Operation
0	0	$P_i = P_{i-1}/2$
0	1	$P_i = (P_{i-1} + B)/2$
1	0	$P_i = (P_{i-1} - B)/2$
1	1	$P_i = P_{i-1}/2$

Anfangswerte: $P_{-1} = 0$, $a_{-1} = 0$
Beim letzten Schritt entfällt die Division durch 2.