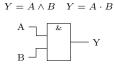
Digitaltechnik

Andrej Scheuer ascheuer@student.ethz.ch 26. Dezember 2020

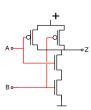
Gates

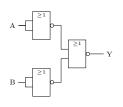
AND



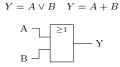
A	В	Y
0	0	0
0	1	0
1	0	0
1	1	1

AND aus NOR



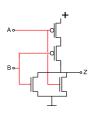


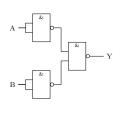
OR



A	В	Y
0	0	0
0	1	1
1	0	1
1	1	1

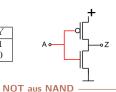
OR aus NAND







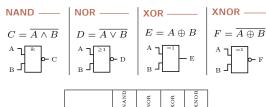








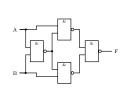
Weitere Gates



		NANI	NOR	XOR	XNOI
A	В	C	D	E	F
0	0	1	1	0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1

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

XOR aus NAND ---



XOR aus NOR: Gleiches Schema wie NAND + 1 Inverter

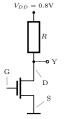
XNOR aus NAND: Gleiches Schema wie XOR aus NOR

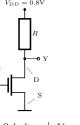
XNOR aus NOR: Gleiches Schema wie XOR $aus\ NAND$

Es versteht sich natürlich, dass wenn von "Gleichem Schema wie..." gesprochen wird, die

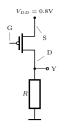
CMOS

NMOS ----PMOS





G	Schalter	Y
0	offen	1
1	zu	0



G	Schalter	Y
0	zu	1
1	offen	0

Konstruktion von CMOS-Gates

Regeln für CMOS-Schaltungen

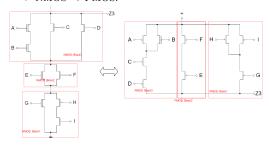
- 1. CMOS-Gates bestehen aus gleich vielen NMOS und PMOS.
- 2. m Eingänge: m NMOS und m PMOS.
- 3. NMOS in Serie \rightarrow PMOS parallel
- 4. NMOS parallel \rightarrow PMOS Serie

Allg. Aufbau CMOS



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.
 - $PMOS \rightarrow NMOS$
- 4. Falls Pull-up:
 - Von V_{DD} aus mit äusserstem Block beginnen.
 - NMOS \rightarrow PMOS.



Funktionsgleichung -

parallel: V Pull-Up: y = 1alle $I: 0 \to I$ invert. Serie: ∧ Pull-Down: y = 0 alle $I: 1 \to Gl$. 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 -

$$(A \land B) \lor (A \land C) = A \land (B \lor C)$$
$$(A \lor B) \land (A \lor C) = A \lor (B \land C)$$

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	$A \lor A = A$	$A \wedge A = A$
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.	$A \lor (A \land B)$	= A
	$A \wedge (A \vee B)$	=A
Nachbar.G.	$(A \wedge B) \vee (\overline{A})$	$\overline{A} \wedge B) = B$
	$(A \vee B) \wedge (\overline{A})$	$\bar{A} \vee B) = B$

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. $\rightarrow 2^n$ mögl. Minterme.	n Schaltvar. $\rightarrow 2^n$ mög 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	В	Y	Minterme	Maxterme
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. \rightarrow 1

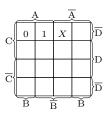
KNF $Y = (A \vee \overline{B}) \wedge (\overline{A} \vee B)$ 1 Maxt. erf. \rightarrow 0

NAND/NOR Schaltungen -

Schaltung nur aus:

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

Karnaugh Diagramme (KVD)

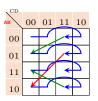


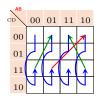
\ CD				
AB	00	01	11	10
00	0	1	X	
01				
11				
10				

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.

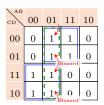
- KVD ausfüllen.
- Päckchen mit 1 uo X.
- 3. Vereinfachte Minterme 3. Vereinfachte Maxterme aufstellen.
- 4. Minterme mit OR ver- 4. Maxterme mit AND binden.

KNF ----

- KVD ausfüllen.
- 2. Päckchen mit 0 uo X.
- aufstellen.
- 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

- zu berechnende positive Zahl
- Basis/Radix von D
- Koeffizient

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

Darstellung D in Basis $R: \ldots b_2b_1b_0.b_{-1}b_{-2}\ldots B$

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

Umwandlung Zahlensysteme

1. Ganzzahlige Division mit R: $D/R = Q_0 + r_0$.

$$Q_i/R = Q_{i+1} + r_{i+1}$$

bis $Q_i = 0$.

3. Erste Operation gibt MSB, letze Operation gibt LSB (aka. unten nach oben lesen.)

Für $1 > D \ge 0$ ——

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

 $a_{-1} \cdot R = P_{-1} \dots$

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

Dezimal zu Binär ---

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	0100 0101 0110 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.

2^{er}Komplement zu Dezimal ———

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

$$D = \sum_{-\infty}^{\infty} b_i \cdot R^i$$
 Wertebereich 2^{er} -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 2^{er}Komp. Übertrag von MSB ignorieren.

Multiplikation -

- Bitweise Multiplikation des Multiplikanden a mit b_i des Multiplika-
- Sukzessive Multiplikationen werden $+b_1 \cdot a \ 0$ um ein Bit (0) nach links verscho- $+b_2 \cdot a \ 0 \ 0$ ben.
- Anzahl Nachkommabits ergibt $+b_3 \cdot a \ 0 \ 0 \ 0$ sich aus der Summe der Anzahl Nachk.bits der Operatoren.

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
- 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
Even r _E	gerade	0	gerade
Odd P_O	ungerade	0	ungerade
Odd P_O	gerade	1	ungerade

01010 11011 10111 00101 00011

Fehler P_E -

Korrekt PE ----

0	1	0	1	0	0	1	0	1	0
			1		1	1	1	1	1
			1		1	0	1	1	1
			0		0	0	1	0	1
0	0	0	1	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 → Änderungen am Eingang können während der ganzen aktiven Taktphase den Output beeinflussen.

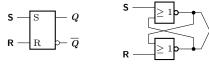
FlipFlops

Taktflankengesteuerte Schaltung → 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 -

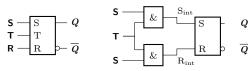


setzt Q auf 1 \mathbf{R} Reset \rightarrow setzt Q auf 0

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

Fall	\mathbf{s}	\mathbf{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 -

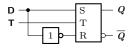


Datenspeicherung Normales SR-Latch

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

D-Latch ---





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

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

→ alter Ausgang gespeichert \rightarrow Input übernommen



FlipFlops





Input beim Übergang von $\mathbf{0} \to \mathbf{1}$ von CLK wirksam.

Input beim Übergang von $1 \rightarrow 0$ von CLK wirksam.

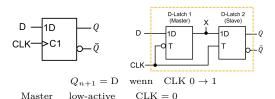


Positive/steigende Takt-flanke

high-active

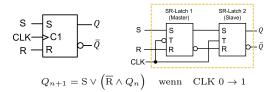
Negative/fallende Takt-flanke

D-FlipFlop

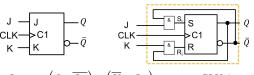


CLK = 1

Slave SR-FlipFlop



JK-FlipFlop



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

Fall	J	\mathbf{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)

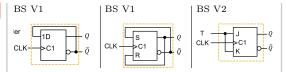
T-FlipFlop -

 $\underline{V1}$ Ausgang wechselt bei jeder aktiven Taktflanke.



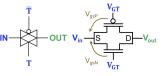
 $Q_{n+1} = \overline{Q_n}$ wenn CLK $0 \to 1$

 $Q_{n+1} = \overline{Q_n}$ wenn CLK $0 \to 1 \land T = 1$



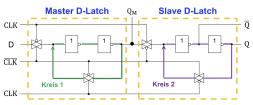
D-FlipFlop in CMOS-Technik

Transmission Gates



IN	\mathbf{T}	Widerstand	OUT
0	0	hochohm.	-
0	1	niederohm.	0
1	0	hochohm.	-
1	1	niederohm.	1

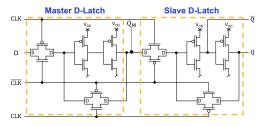
TG sperrt wenn Widerstand hochohmig ist. (T = 0)



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 $(F_1 \wedge \overline{Q_n}) \vee (F_2 \wedge Q_n)$ geschrieben werden kann.

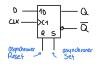
Gleichung für D-FF ightarrow JK-FF

- 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.

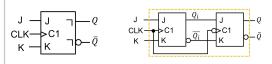
 t_{vd} Verzögerungszeit Durchlaufzeit

$$T_{\min} \ge t_{\mathrm{pd1}} + t_{\mathrm{pd,ks}} + t_{\mathrm{s2}}$$
 $f_{\max} = \frac{1}{T_{\min}}$

 t_h kann bei der Berechnung von $f_{\rm max}$ vernachlässigt werden

Es wird der längste PFad zwischen zwei FlipFlops betrachtet.

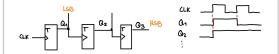
Zwischenspeicher-FF -



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.

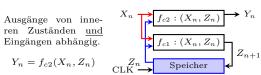
Bei synchronen Automaten besitzen alle Speicherelemente (FlipFlops) den gleichen Takteingang.

Formale Beschreibung

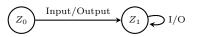
$$X=(x_1,\ldots,x_e) \\ Y=(y_1,\ldots,y_b) \\ Z=(z_1,\ldots,z_m) \\ Z_0\in Z \\ f_{c1}:(X_n,Z_n)\to Z_{n+1} \\ f_{c2}:(X_n,Z_n)\to Y_n \\ Eingangsalphabet mit entite box Ausgängen Mit box Ausgängen Zustandsmenge mit minternen Zuständen Anfangszustand Übergangsfunktion Ausgangsfunktion Ausgangsfunktion$$

Automatentypen

Mealy-Automat -



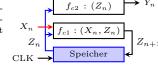
Zustandsdiagram



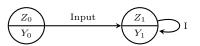
Moore-Automat

Ausgänge nur von inneren Zuständen abhängig (keine Verbindung zwischen Input und Output).

 $Y_n = f_{c2}(Z_n)$



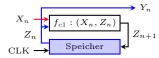
Zustandsdiagram



Medwedjew-Automat -

Ausgänge entsprechen inneren Zuständen.

 $Y_n = Z_n$



Zustandsdiagram



Nachtrag Zustandsdiagram -

Knoten interne Zustände

Kanten Übergänge zwischen Zuständen

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

Zustandsfolgetabelle

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

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

 $e + 2m + b$ Spalten 2^{e+m} Zeilen

Wichtig: für e, m, b Anzahl Bits verwenden, nicht Anzahl Zustände.

Entwurf eines Automaten

- 1. Auftrag lesen und analysieren \rightarrow Automatentyp bestimmen.
- Zustandsmenge bestimmen → Anzahl erforderlich D-FlipFlops [log₂(Anzahl Zustände)].
- Eingangs- und Ausgangsvariablen definieren, Kodierung.
- 4. Darstellung der Zustandsfolge in einem Zustandsdiagram.
- 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 \rightarrow neu kodieren.
- 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 Störungsan-

fällig)

Asynchronzähler

Dualzähler Kaskadierung von T-FlipFlops Vorwärtszähler negativ flankengesteuerte Flip-Flops

1 " ' " " 1 7

Rückwärtszähler $\overline{Q_i}$ benutzen oder positive flankengesteuerte FlipFlops.

- Anzahl Bits = Anzahl T-FlipFlop
- LSB nach 1. FlipFlop, MSB ganz rechts

Probleme von Asynchronzählern -

- Verzögerungen der Zustandsänderungen kumulieren sich entlang der Schaltung.
- Zeitverzögerung ist bei jedem Zustand anders.
- Damit jeder mögliche Zustand bei n Flip Flops (kurz) auftritt:

$$f_{max} = \frac{1}{\sum_{i=1}^{n} t_{pd,i}}$$

Modulo-n Zähler

Zählt bis zu einem bestimmten Zustand und springt dann auf einen definierten Zustand zurück. Es werden n Zustände durchlaufen.

Kombinatorische Schaltung (AND-Gates) registrieren den Endzustand und setzen den definierten Zustand mittels der asynchronen Set- und Reset-Eingänge.

Def. Z (Bit) $\begin{pmatrix} 0 & R \\ 1 & S \end{pmatrix}$ mit komb. Schalt. verbinden

Anderer asynchroner Eingang an GND.

Synchronzähler

Alle FlipFlops haben das selbe Taktsignal. Meistens **Medwedjew**-Automaten.

Entwurf

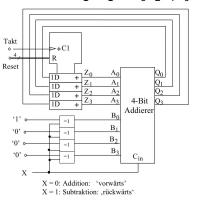
- 1. Zustandsgraph zeichnen.
- 2. Folgezustandstabelle aufstellen.
- 3. Für alle Folgezustände KV-Diagramme erstellen \rightarrow Gleichung Folgezustand.
- 4. Zeichnen (Ausgänge = interne Zustände)

Vorwärts-Rückwärtszähler

Zusätzlicher Eingang bestimmt Zählrichtung \rightarrow wie Synchronzähler entwerfen.

Alternative

Zähler Ausgänge: Z₃Z₂Z₁Z₀



D-FlipFlops mit Addierer kombinieren.

$$\begin{array}{cccccc} X & B_3B_2B_1B_0 & & & & & & & & & & & & & & & & \\ 0 & & 0001 & & \rightarrow & & +1 & & & & & & & \\ 1 & & 1111 & & \rightarrow & & -1 & & & & & & & & & \end{array}$$
 addiert

Diverses

Physikalische Zuordnung logischer Zustände

0 Low 0 V Ground 1 High 0.8 V VDD

Toleranzen:

- GND: 0 V... 0.15 V
- VDD 0.7 V... 0.9 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 ${\cal A}$ und ${\cal B}.$ Produziert Summe und Carry-Out.

$$SUM = A \oplus B$$
 $CO = A \wedge B$

Volladdierer --

Nimmt einen zusätzlichen Input CI entgegen.

$$SUM = (A \oplus B) \oplus CI$$
 $CO = (A \land B) \lor (S_{AB} \land CI)$

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.
- → Schnell aber Schaltungsaufwendig

Ripple-Carry Addierer (Paralleladdierer) -

Vorteile

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

Nachte

• SUM und CO für die i-te Stelle können erst nach der Berechnung der (i – 1)-ten Stelle gebildet werden.

Nachteile Bei Addition

von n-stelligen Summan-

den müssen $\sim n \cdot 2^{2n-1}$

ver-

Min-/Maxterme

knüpft 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 \to 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
Ī	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.