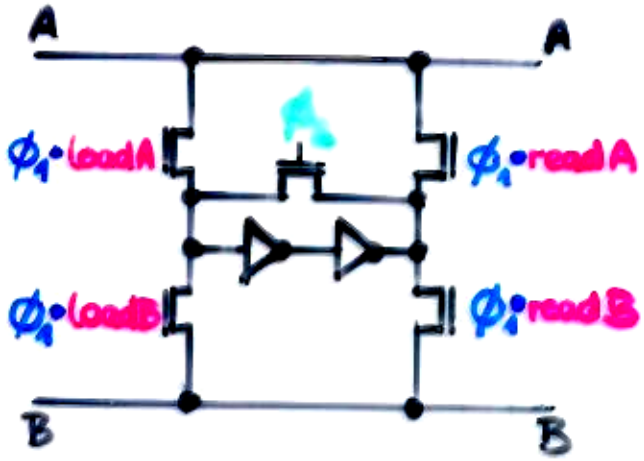
REGISTER

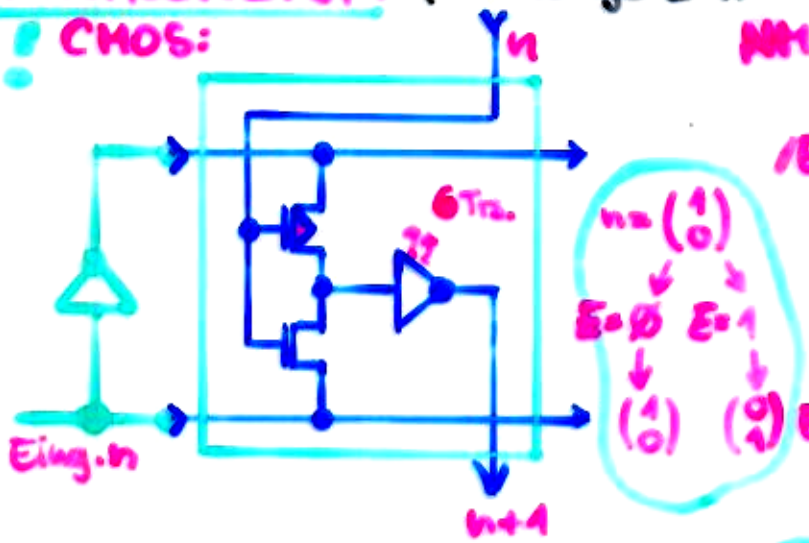
MC80: Accu:



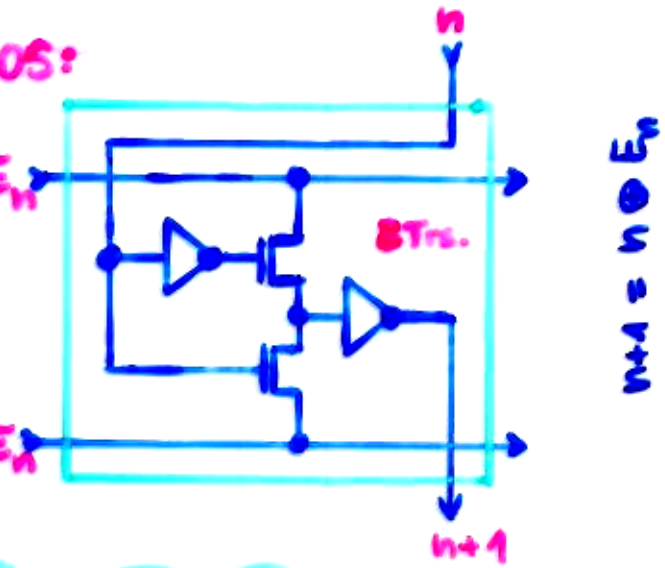
Paritätsprüfung: Antivalenz

ANTIVALENZTYP: (Kettenlänge ≥ 4)

CMOS:



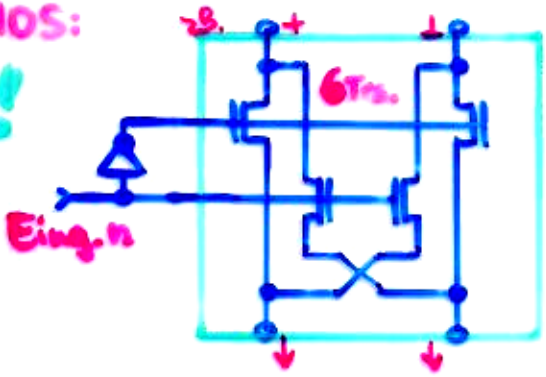
NMOS:



n+1 = n ⊕ En

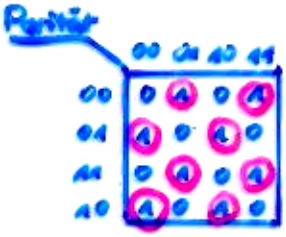
SCHALTTYP: (Kettenlänge ≤ 8)

NMOS:



Problem: keine Termkürzung in Karnaughtafel

z.B.: 4 Eingänge (n=4) für 4 Zellen:

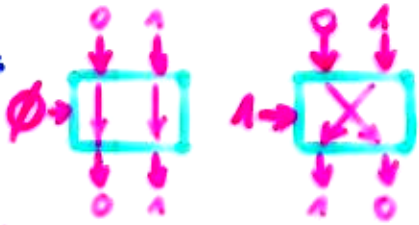


Aequivalenz

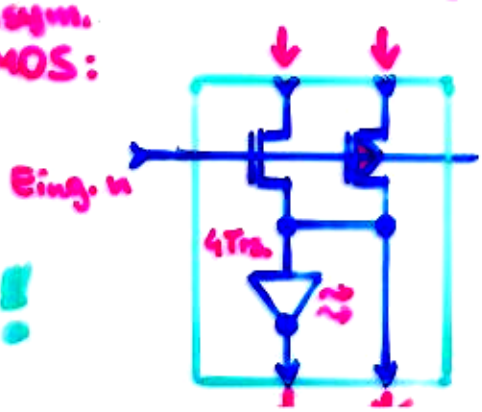
Klassische Realisierung:

2^{n-1} - Gatter mit je n Eingängen!

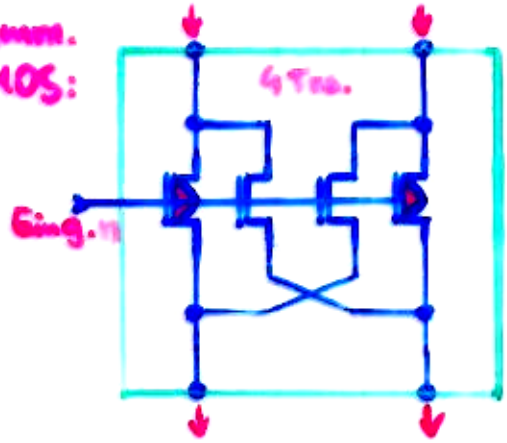
Prinzip:



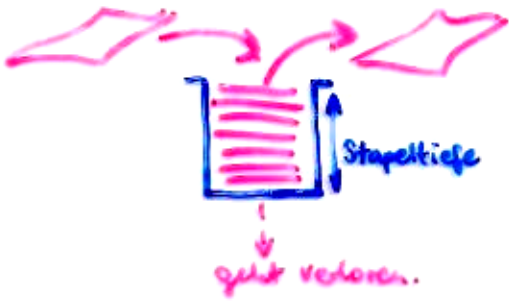
unsym. CMOS:



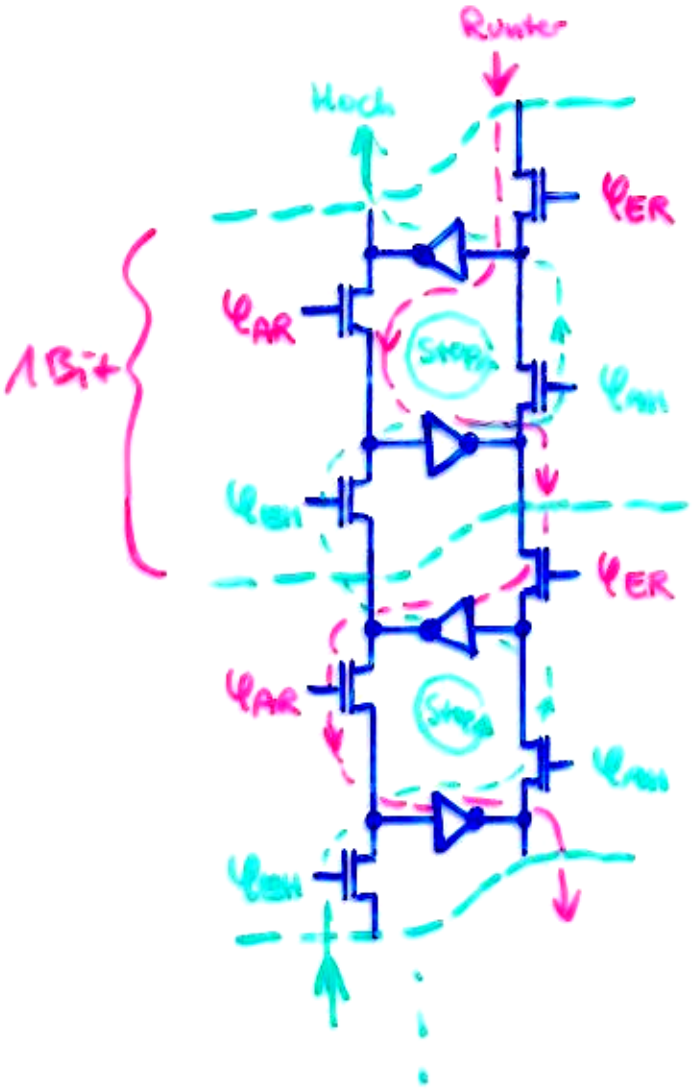
symm. CMOS:



STAPELSPEICHER (STACK, LIFO [last in - first out])



Statischer STACK:



Ansteuerung: ©Gi

Stop (S): $S = \overline{R \vee H} = \bar{R} \cdot \bar{H}$

$\| \varphi_{AR}, \varphi_{AH} = H$ (speichern)

$\varphi_{ER}, \varphi_{EH} = L$ (Sperren)

Runter (R):

$\| \varphi_{ER} = \varphi_E$, $\| \varphi_{AR} = \varphi_A$

$\varphi_{EH} = \varphi_{AH} = L$

Hoch (H):

$\| \varphi_{EH} = \varphi_E$, $\| \varphi_{AH} = \varphi_A$

$\varphi_{ER} = \varphi_{AR} = L$

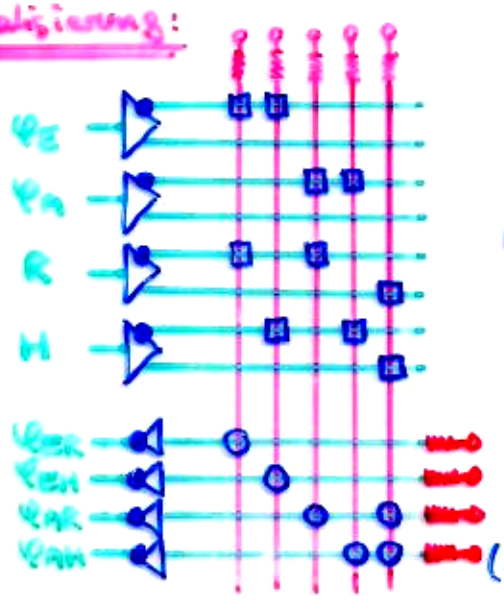
Funktionen:

$\varphi_{ER} = \varphi_E \cdot R$, $\varphi_{EH} = \varphi_E \cdot H$

$\varphi_{AR} = \varphi_A \cdot R \vee \bar{R} \cdot \bar{H}$

$\varphi_{AH} = \varphi_A \cdot H \vee \bar{E} \cdot \bar{H}$

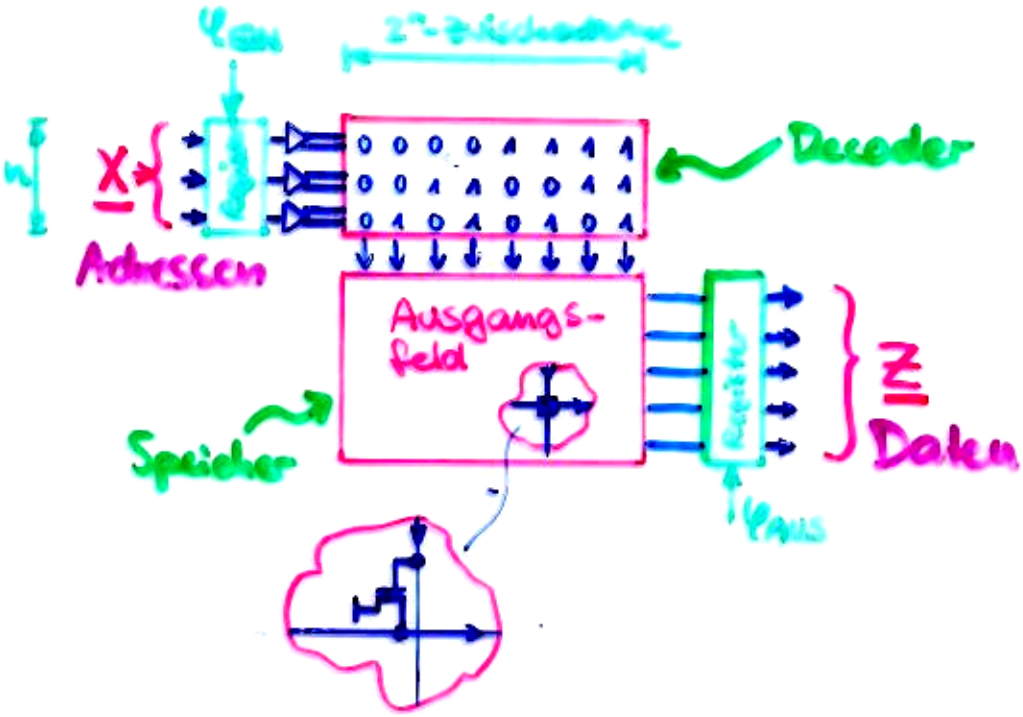
Realisierung:



PLA - Sonderformen:

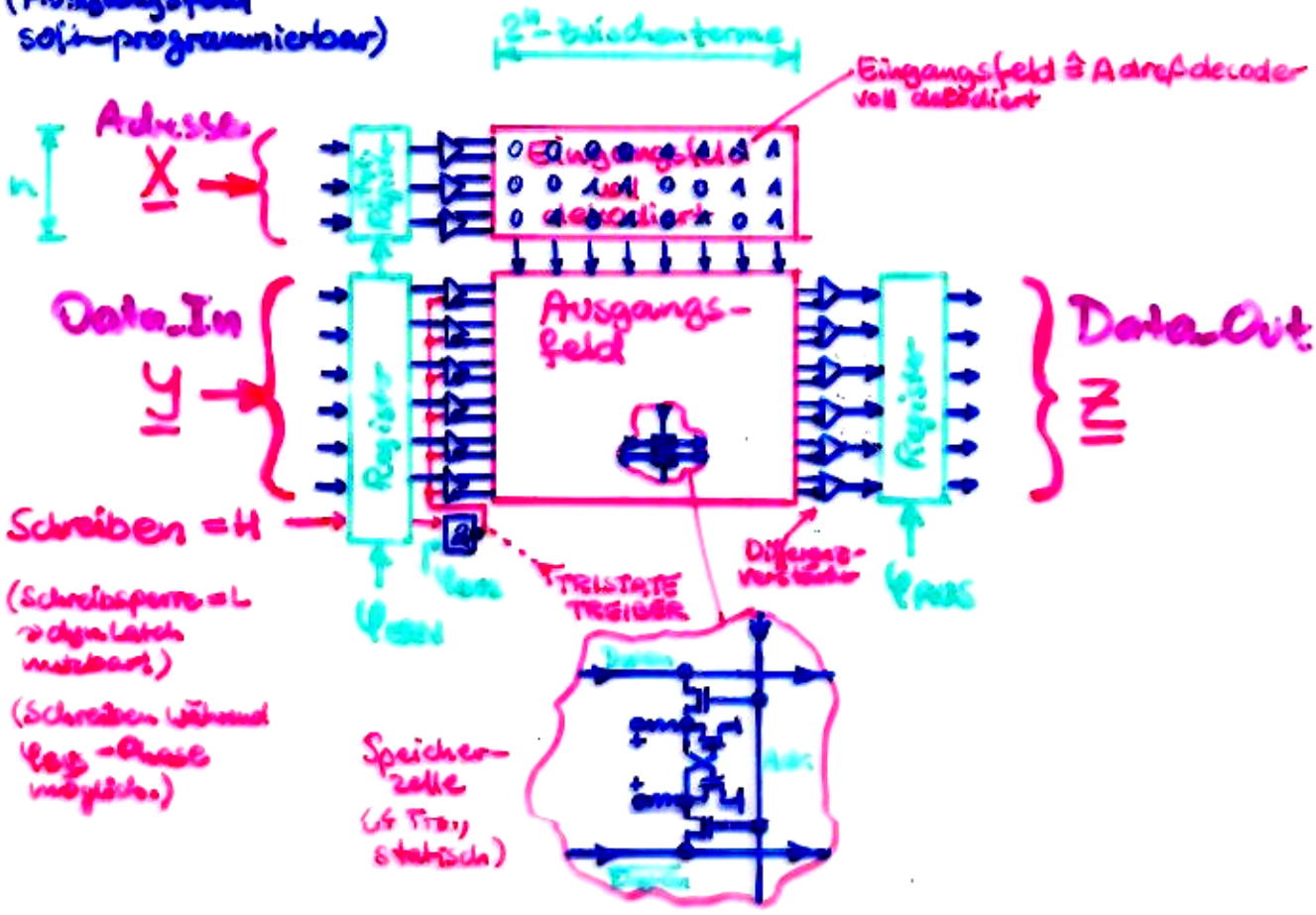
ROM (read only memory)

(Eingangsfeld fest verdrahtet und voll dekodiert, Ausgangfeld „hart-programmierbar")



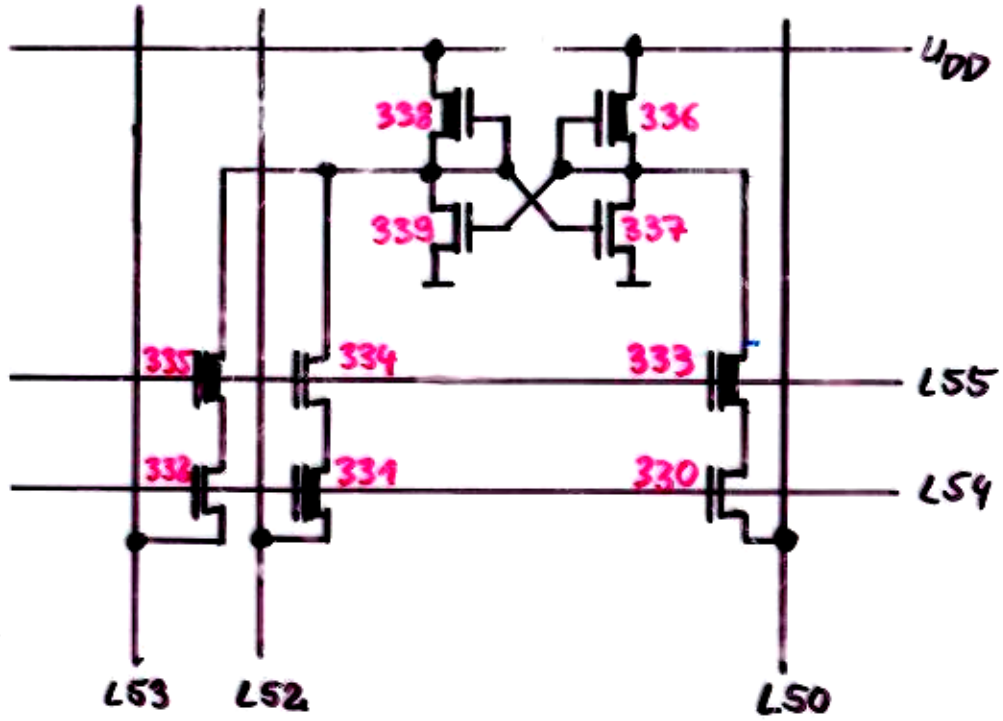
RAM (random access memory)

(Ausgangsfeld soft-programmierbar)

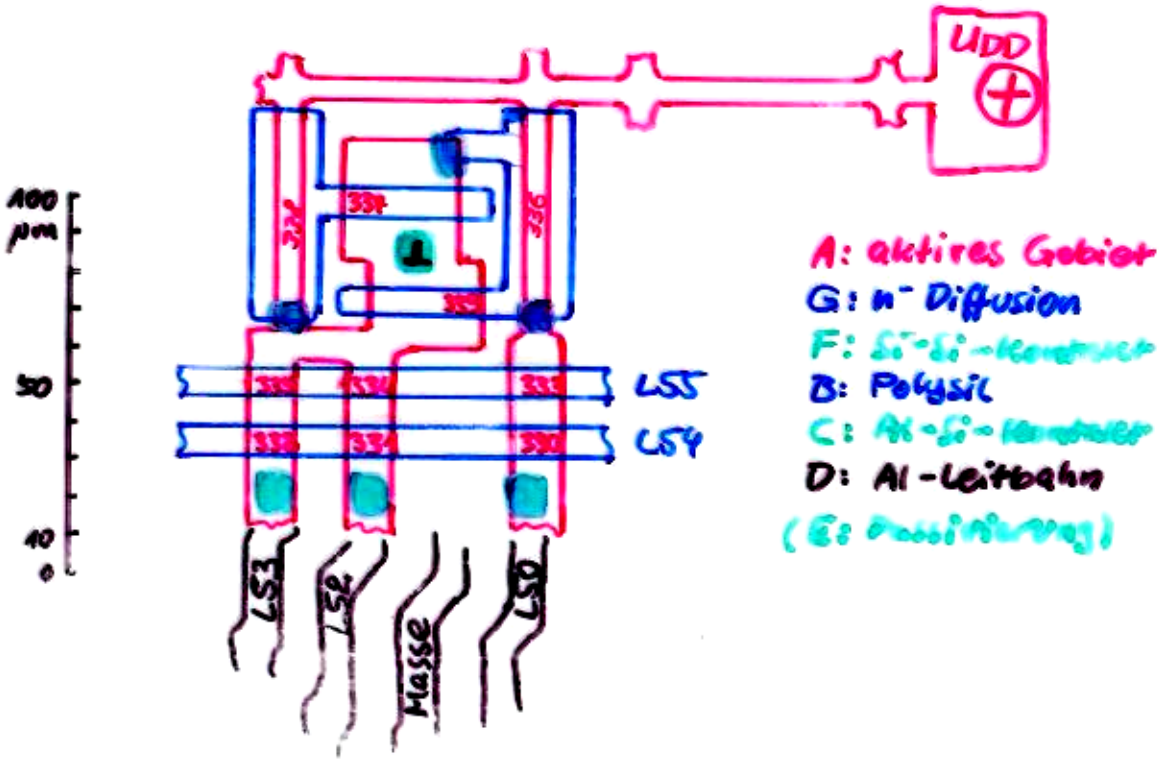


RAM-Speicherzelle mit zusätzlichem Leseausgang

SCHALTUNG:



LAYOUT :

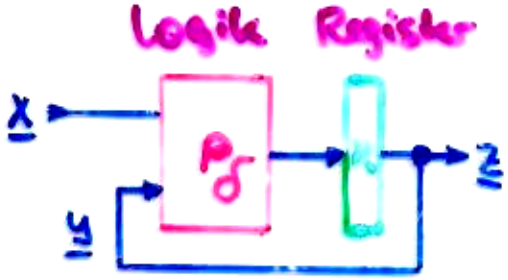


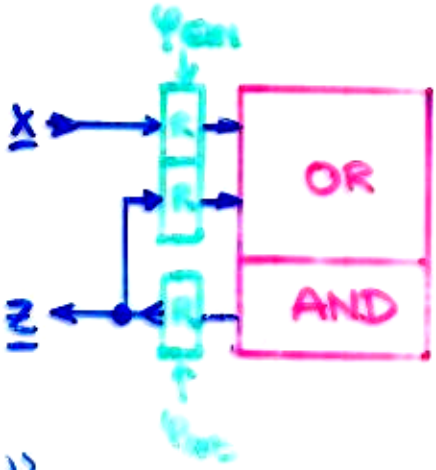
PLA – Automatentypen (mit Speicher)

F: Zuordner (PLA)
R: Speicherregister
δ: Überführungsfkt. $\underline{X} * \underline{y} \to \underline{y}$
λ: Ausgangsfkt. $\underline{X} * \underline{y} \to \underline{z}$

(Initialisierung: $\underline{X}$-Bit kleinmit über einen Zwischenterm alle $\underline{y}$ und $\underline{z}$-Leitungen!)

Medwedjew-Automat:

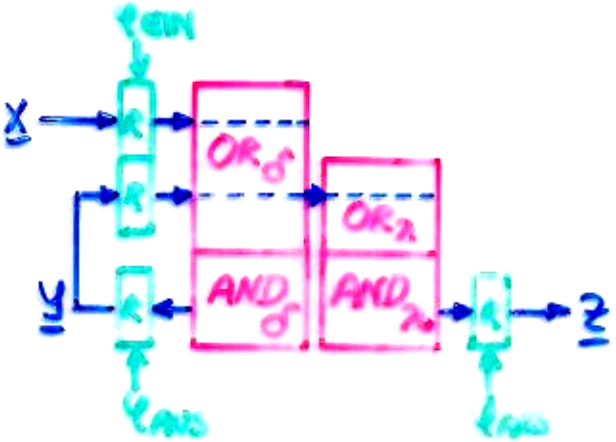




$$\underline{z}^n = \underline{y}^n = \delta(\underline{x}^{n-1}, \underline{y}^{n-1})$$

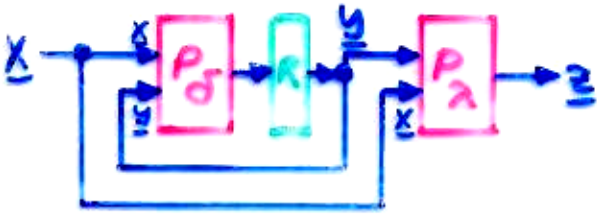
$$\{\underline{z}(t_n) = \underline{y}(t_n) = \delta(\underline{x}(t_{n-1}), \underline{y}(t_{n-1}))\}$$

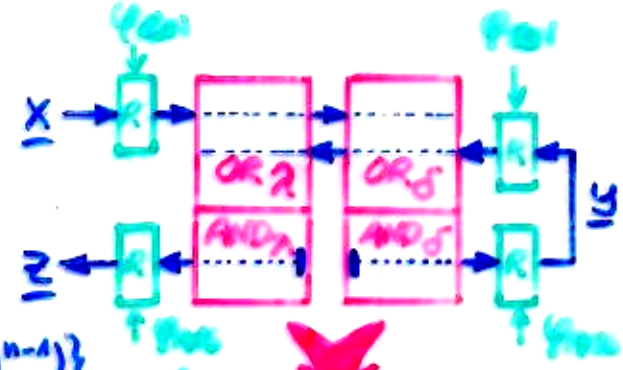
Moore-Automat:



$$\underline{z}^n = \lambda(\underline{y}^n) = \delta(\underline{x}^{n-1}, \underline{y}^{n-1})$$

Mealy-Automat:

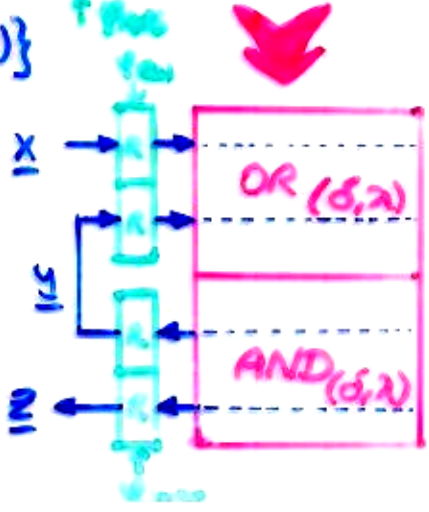




$$\underline{z}^n = \lambda(\underline{x}^{n-1}, \underline{y}^n) = \lambda\{\underline{x}^{n-1}, \delta(\underline{x}^{n-1}, \underline{y}^{n-1})\}$$

$$\uparrow$$

$$\underline{y}^n = \delta(\underline{x}^{n-1}, \underline{y}^{n-1})$$



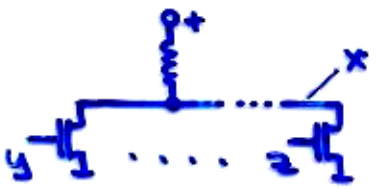
$\underline{X}$: Eingangszustände

$\underline{y}$: Innere Zustände

$\underline{z}$: Ausgangszustände

Normalform - PLA (KDF)

NSGT: vorteilhaft NOR-Logik: $A\downarrow$ $V\uparrow$ $t_v\downarrow$

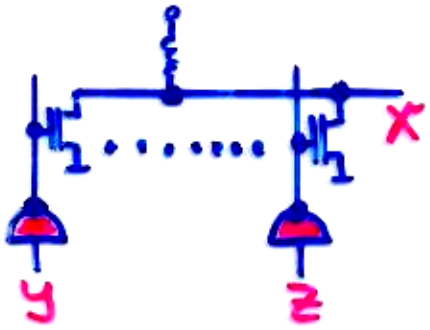


$\bar{x} = y \vee z$ ← ODER

$x = \bar{y} \cdot \bar{z}$ ← UND

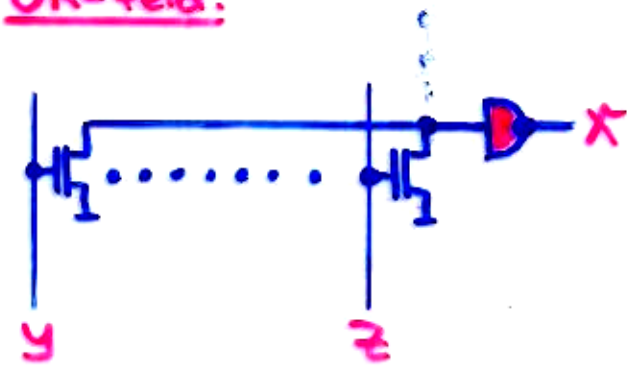
$x = \overline{y \vee z}$ – NOR.

AND-Feld:



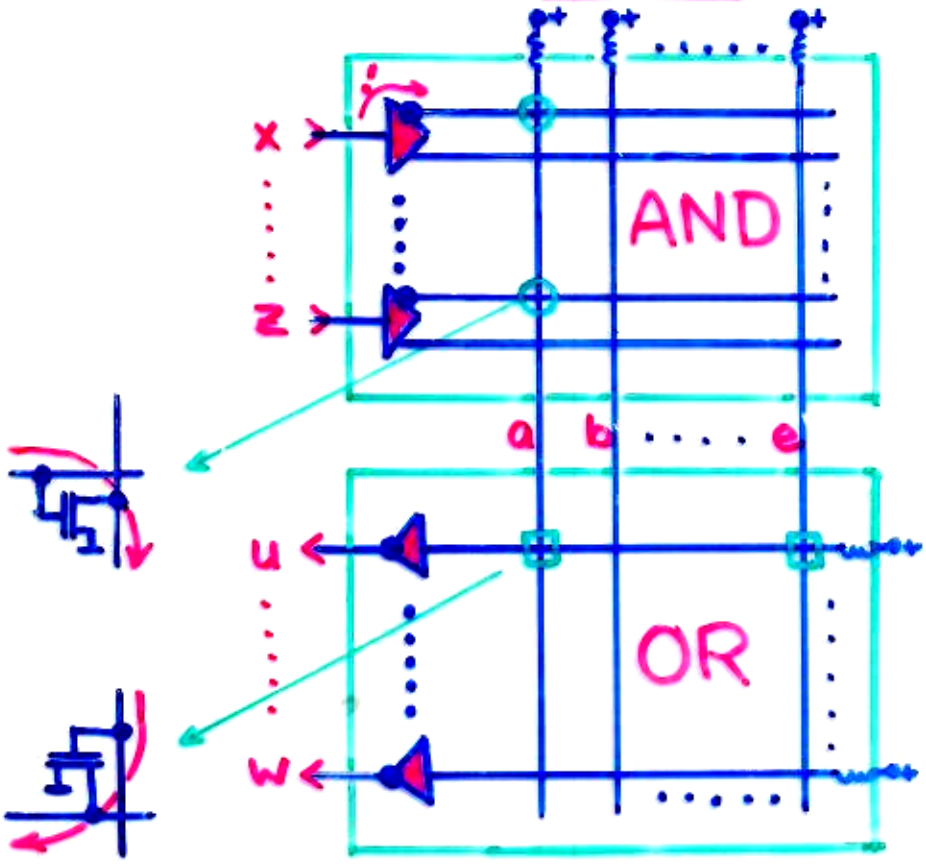
$x = y \cdot z$

OR-Feld:



$x = y \vee z$

KDF - **PLA** (ANDOR)



Beispiel:

$a = x \cdot z$

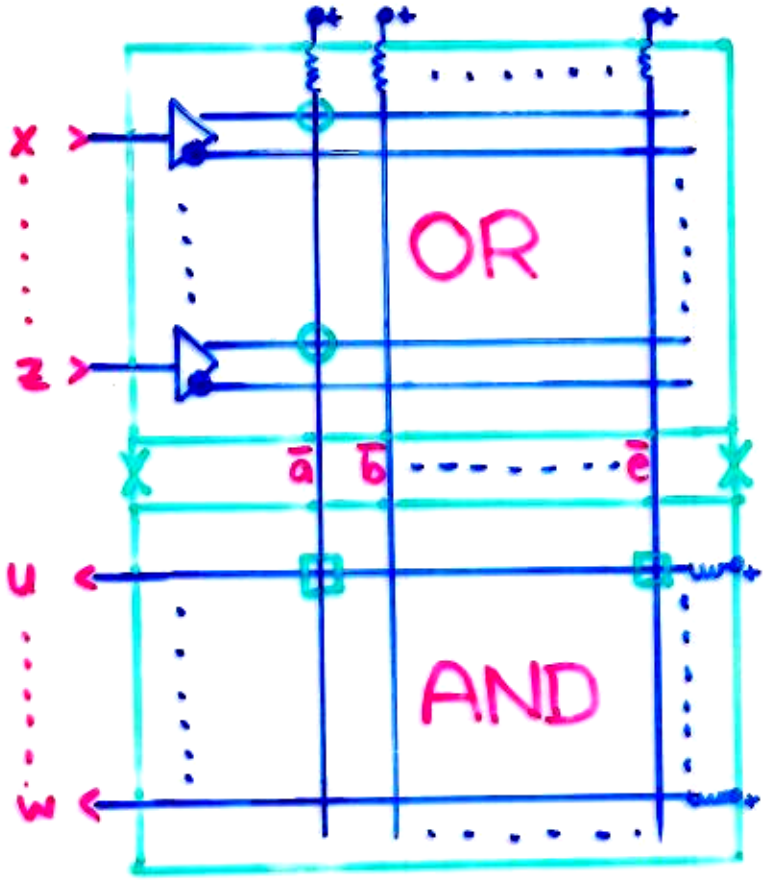
„AND"-Feld

$u = a \vee e$

„OR"-Feld

Hinweis: KDF: Ausgang „high"-aktiv.

KKF - PLA (ORAND)



$\bar{o} = \overline{x \vee z}$

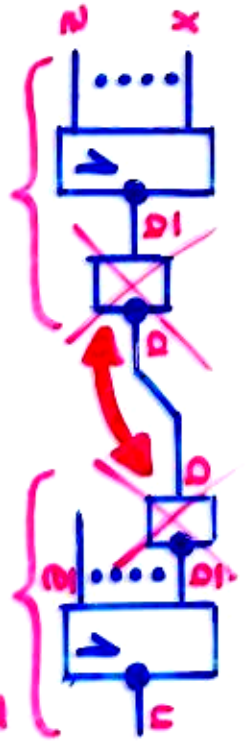
$o = x \vee z$

„OR"-Feld

$u = \overline{\bar{a} \vee e}$

$u = a \cdot e$

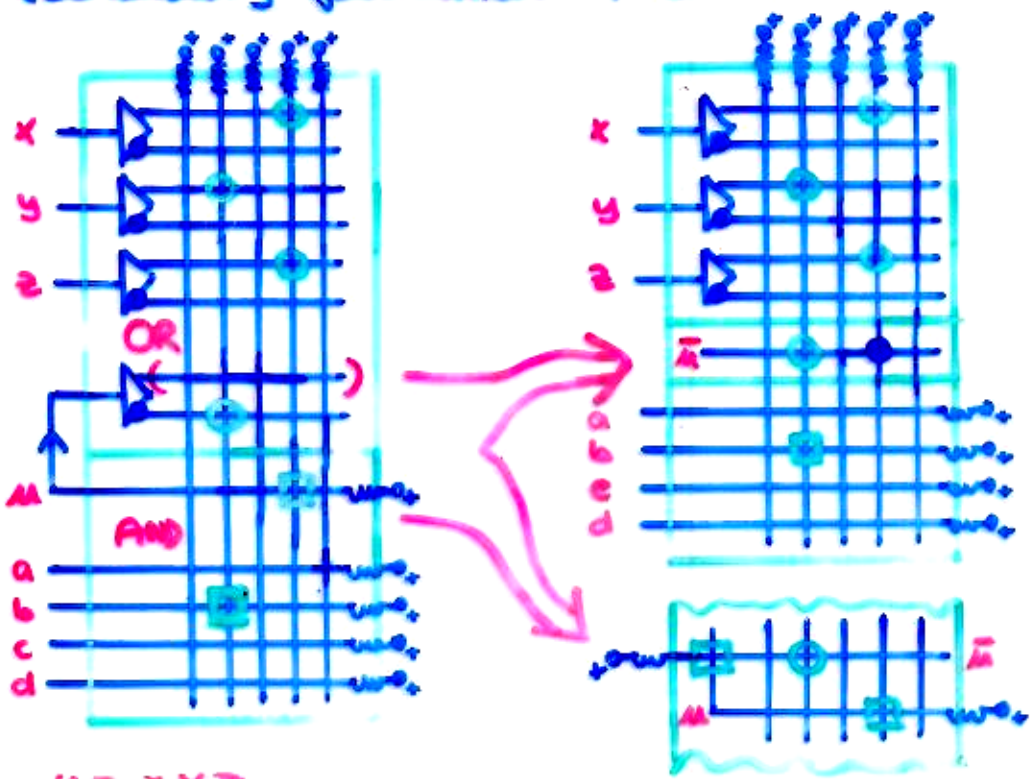
„AND"-Feld



Hinweis: KKF: Ausgang „low"-aktiv

PLA mit Zwischentermfeld

(Behandlung geklammerter Ausdrücke)

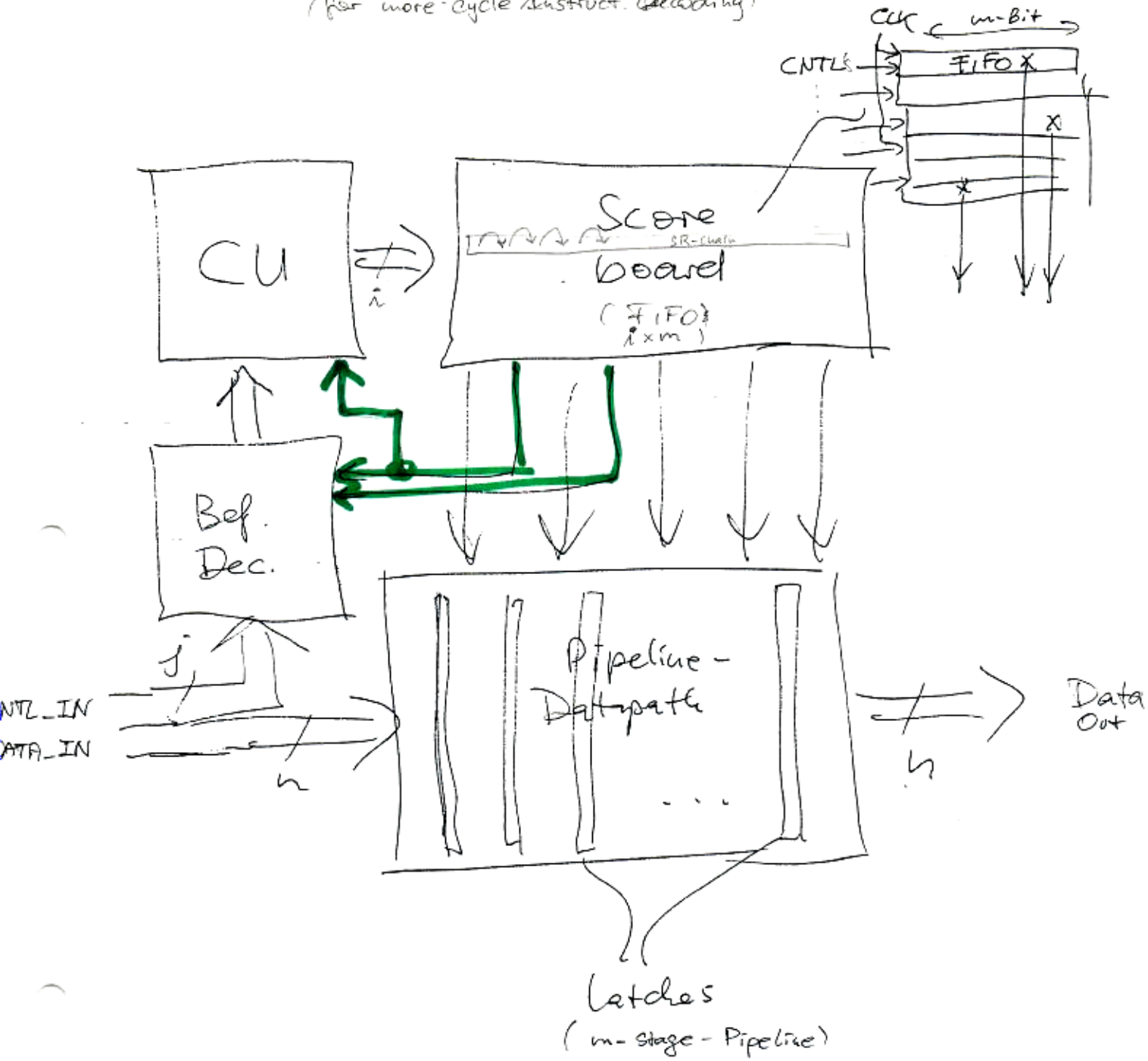


$u = x \vee z$

$b = \bar{u} \vee y = \overline{(x \vee z)} \vee y = (\bar{x} \bar{z}) \vee y$

Pipeline

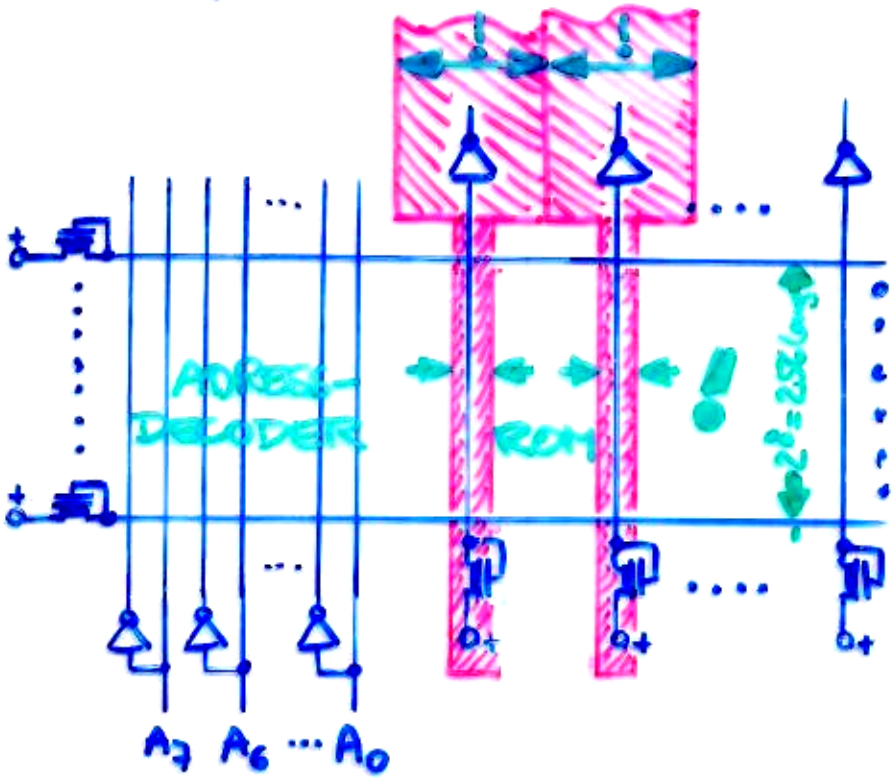
→ : possible, but not beauty
(for more-cycle instruct. Reading)



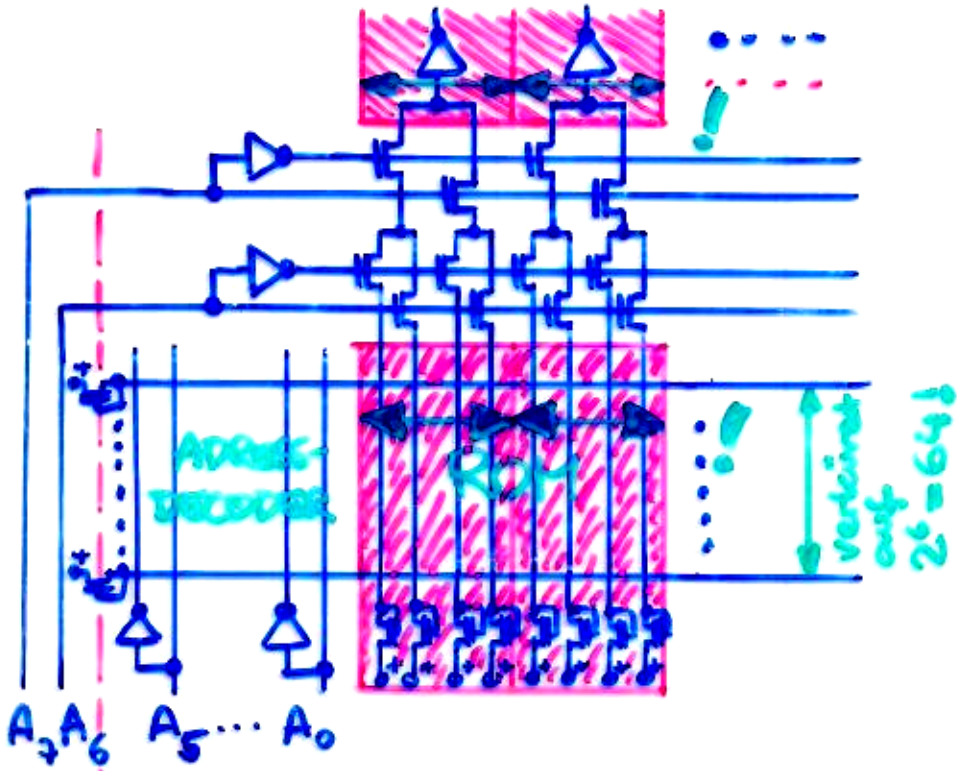
Regularität schaffen!?!

Problem: Regularität contra Flächenausnutzung!

Beispiel: ROM:



Lösung: Speicherorganisation ändern:

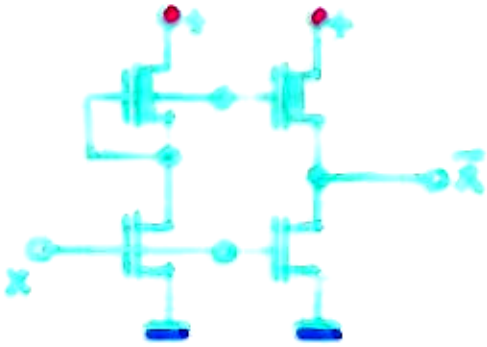


PLANARISIERUNG

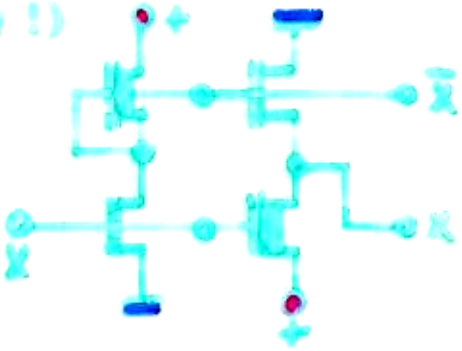
"Kreuzungen in die GND und PLUS - Ugen hineinschieben!"

TREIBER

Invertierend:

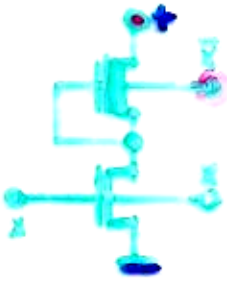


Nichtinvertierend bzw. Rail: ($U_{min.} > 4V$!)

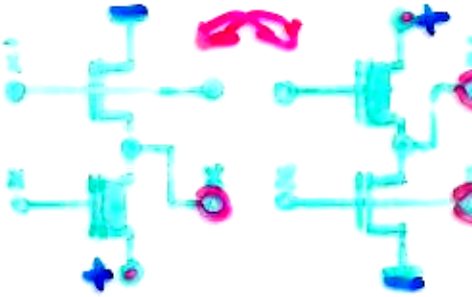


TREIBER-KOMPOSITS:

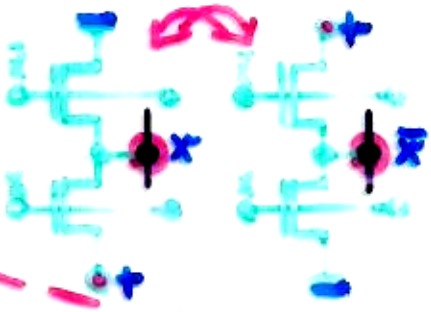
Inverter:



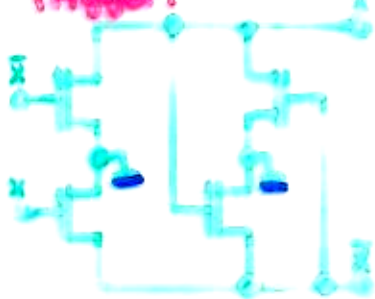
Bootstrap-Inv.:



OE-Treiber:

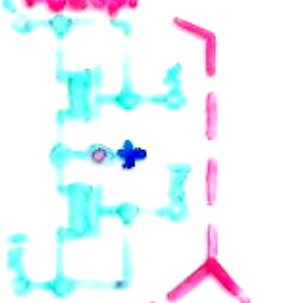


Trigger:

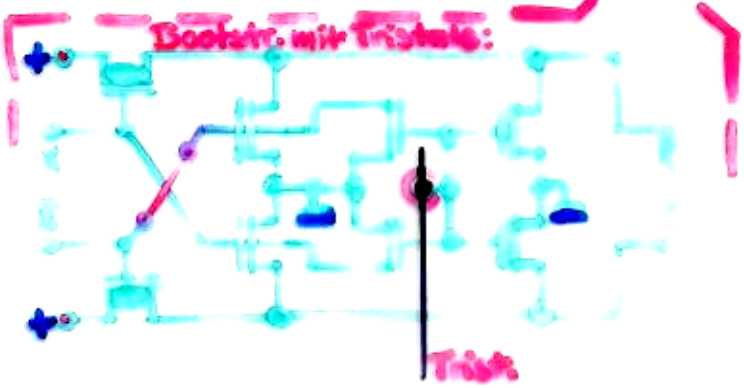


Tristate:

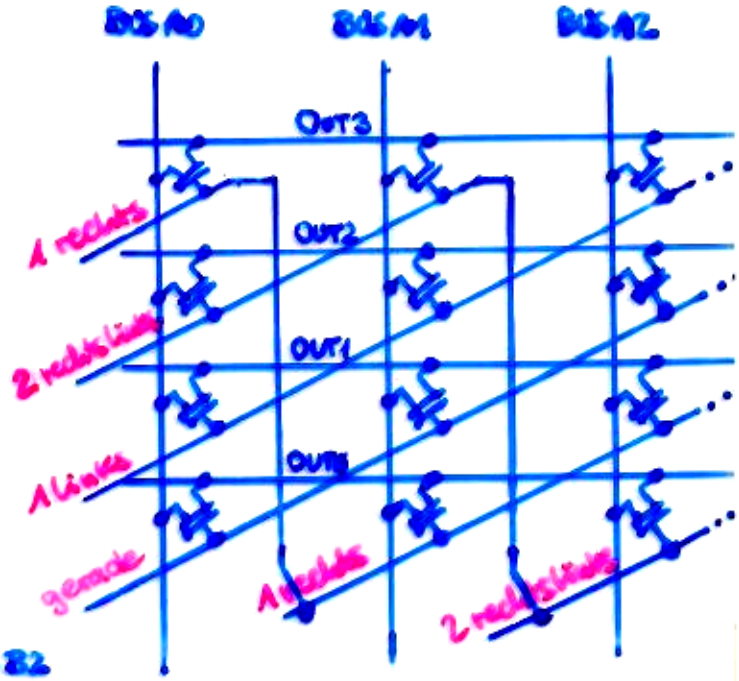
Pullup:

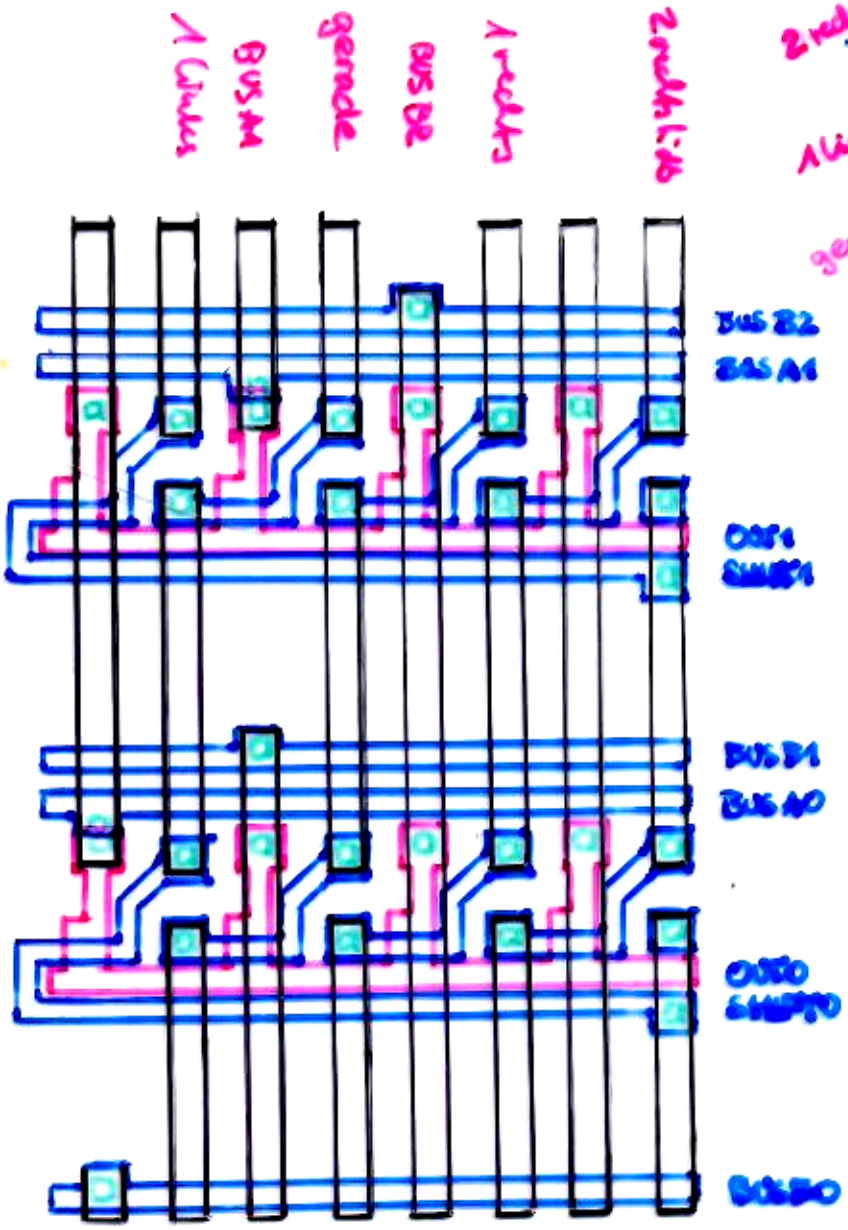


Bootstr. mit Tristate:



Prinzipschaltung:





Lambda-Layout

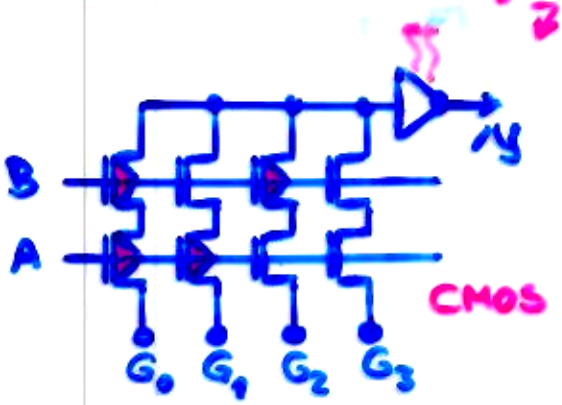
UMLAUF-SCHIEBER

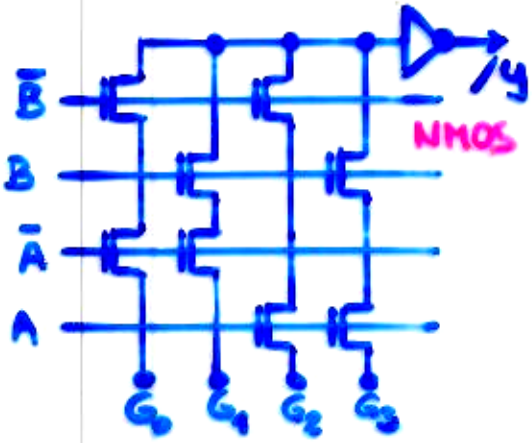
M/C: OM2: BARRELSHIFTER

BOOLSCHER BLOCK

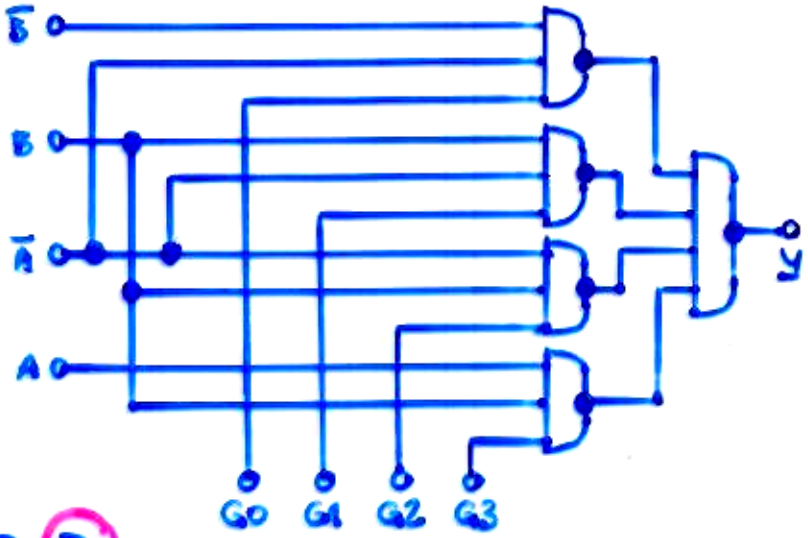
(MC'80: FUNCTION BLOCK)
(ELSTAC: SELEKTOR)

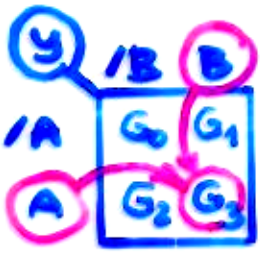
Schaltverealisierungen:





Gatterrealisierung:

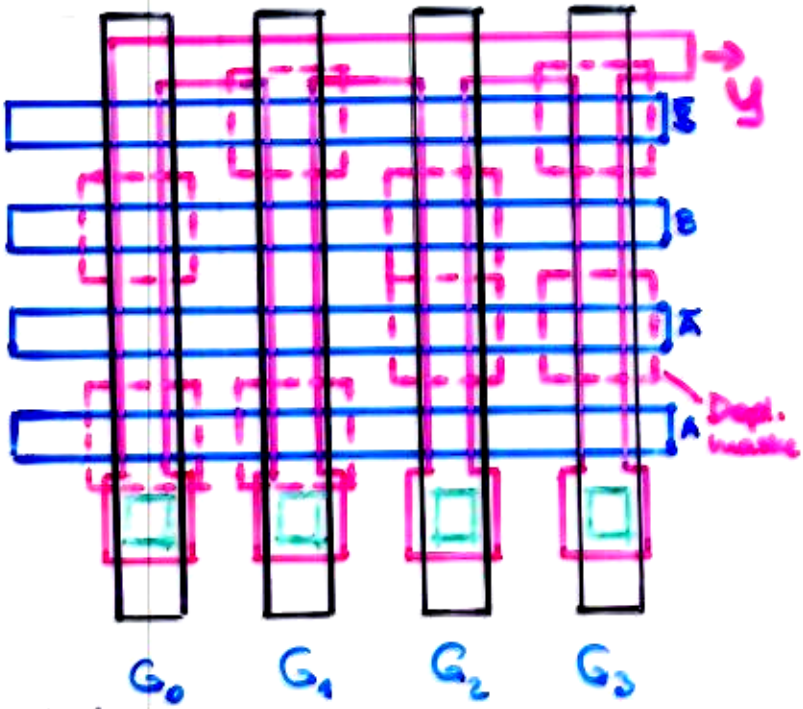




ausgew. Schaltfkt (ALU):

G_0	G_1	G_2	G_3	Funktion	
0	0	0	0	y=0	ZERO
1	1	1	1	y=1	ONE
0	0	1	1	y=A	LD A
1	1	0	0	y=Ā	LD Ā
0	1	0	1	y=B	LD B
1	0	1	0	y=B̄	LD B̄
0	0	0	1	y=AB	AND
1	1	1	0	y=$\overline{AB}$	NAND
0	1	1	1	y=A∨B	OR
1	0	0	0	y=$\overline{A \vee B}$	NOR
0	1	1	0	y=A⊕B	ANT
1	0	0	1	y=$\overline{A \oplus B}$	ÄQ

NSGT-Layout:

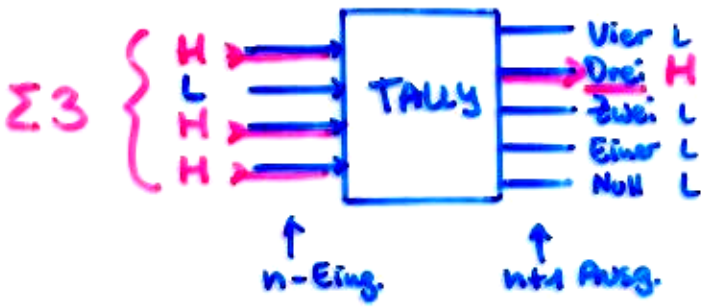


Verhalten:

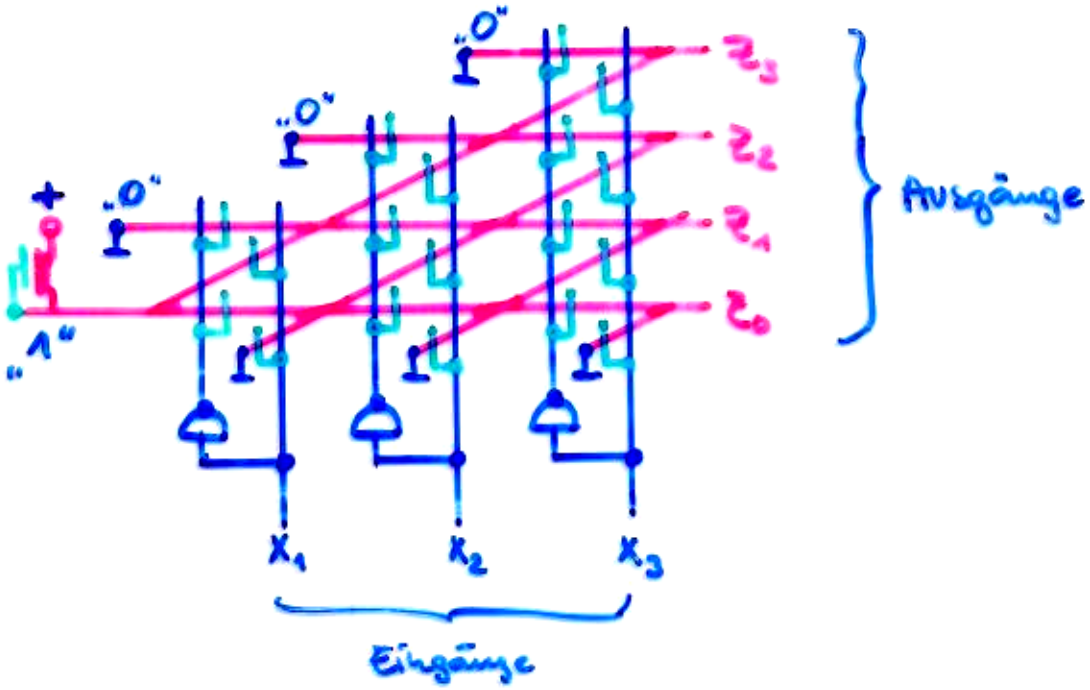
$y = \bar{A}\bar{B}G_0 \vee \bar{A}BG_1 \vee A\bar{B}G_2 \vee ABG_3$

ABZÄHLER (TALLY) (MC S. 78 ff)

Problem: K-Eingänge sind „hoch" → k-ter Ausgang soll „hoch"



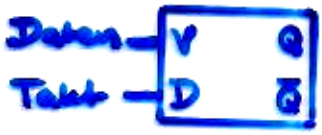
3-er TALLY:

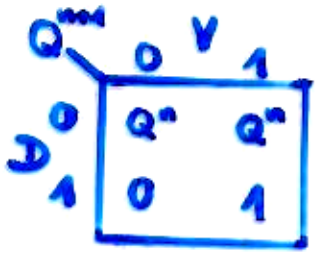


Algorithmus:

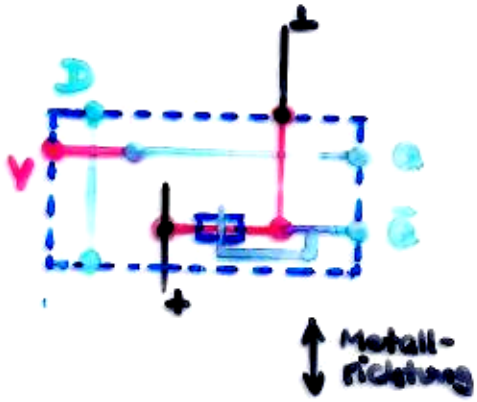
Ein „hoch" speist den Null-Ausgang Z_0 (Pull-up).
Jedes „hoch" eines (jeden) Einganges schiebt diese „1" um eine Etage nach oben.

DV-Latches:

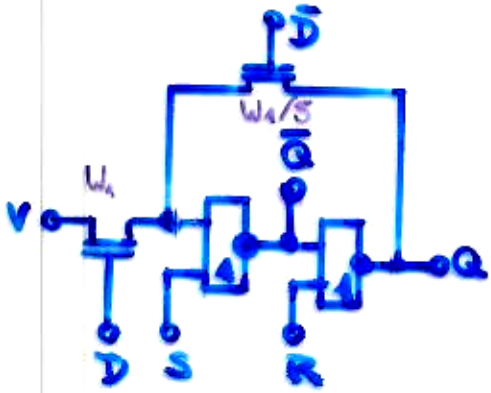


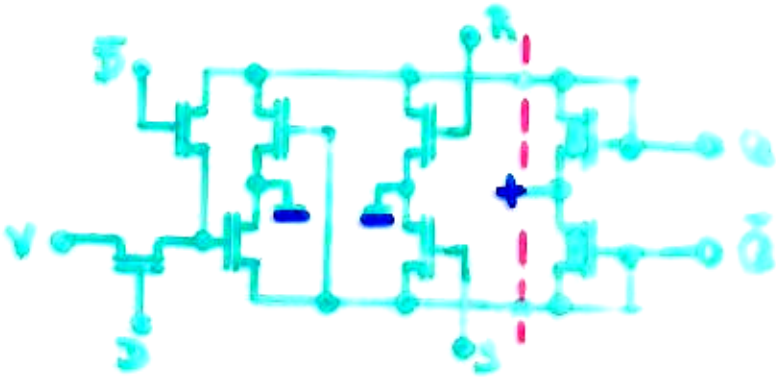


Dyn. DV-Latch:

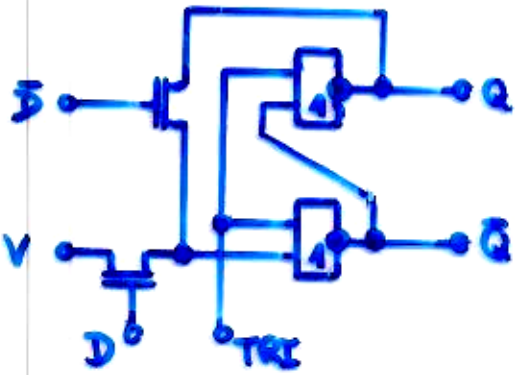


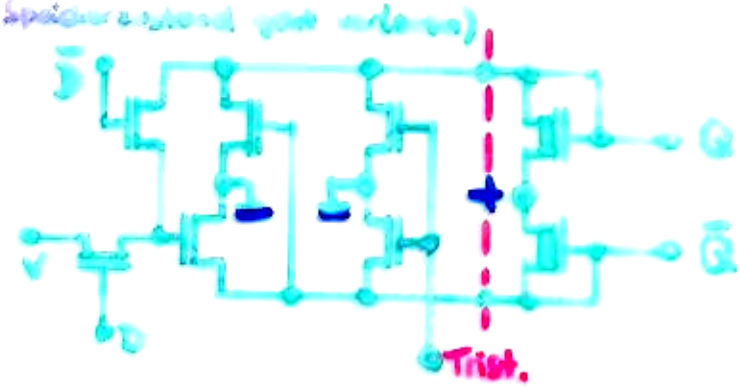
Stat. DVL, set- u. rücksetzb.:





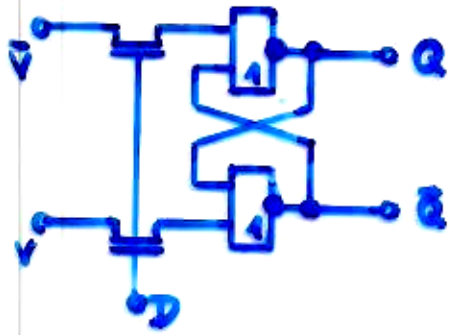
Stat. DVL mit Tristate (Speicherzustand wird erhalten)

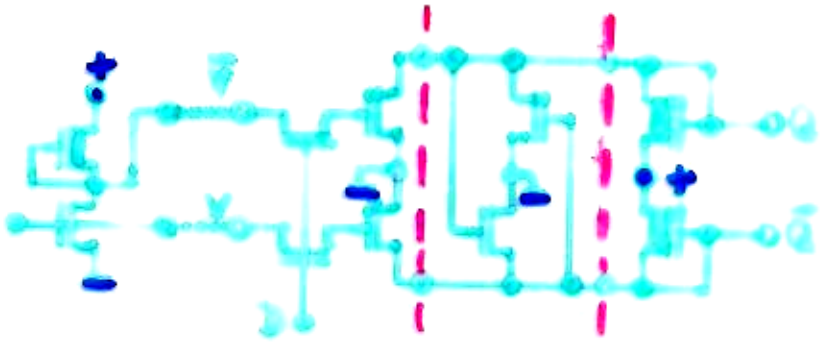




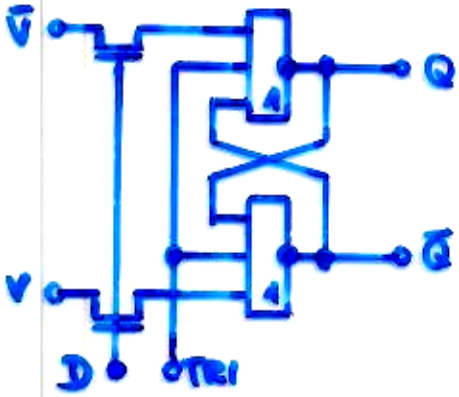
(Zu Planarisierungen, DV-Latches)

DV-Latch, statisch, triggernd:





DV-Latch, statisch, triggernd, mit Tristate (Speicherfunktion wird gelöscht)

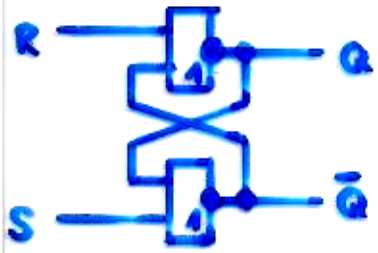


RS-Flipflops

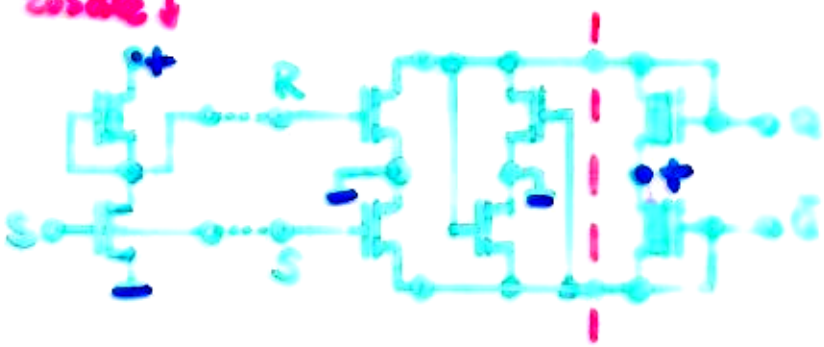
$Q^{n+1}, \bar{Q}^{n+1}$	S = 0	S = 1
R = 0	$Q^n\ \bar{Q}^n$	10
R = 1	01	00

NOR-Typ
Vorteilhaft für Endstufen-Ansteuerung!

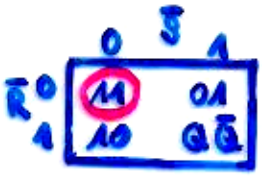
RS-Latch

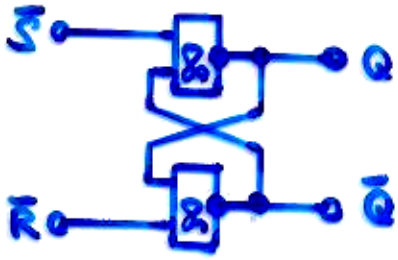


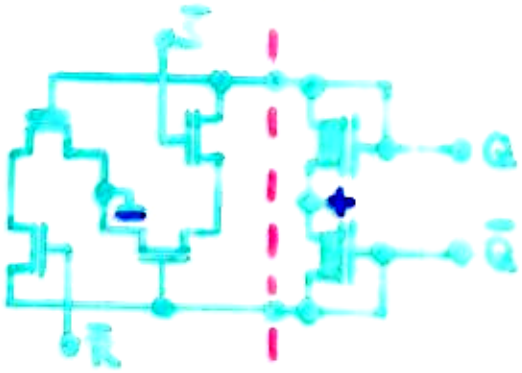
Trigger-Zustände ↓



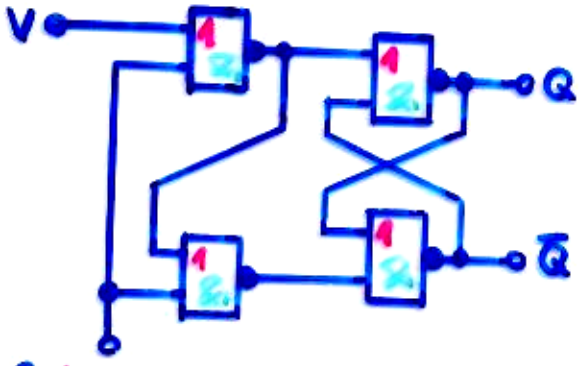
RS-Flipflop:



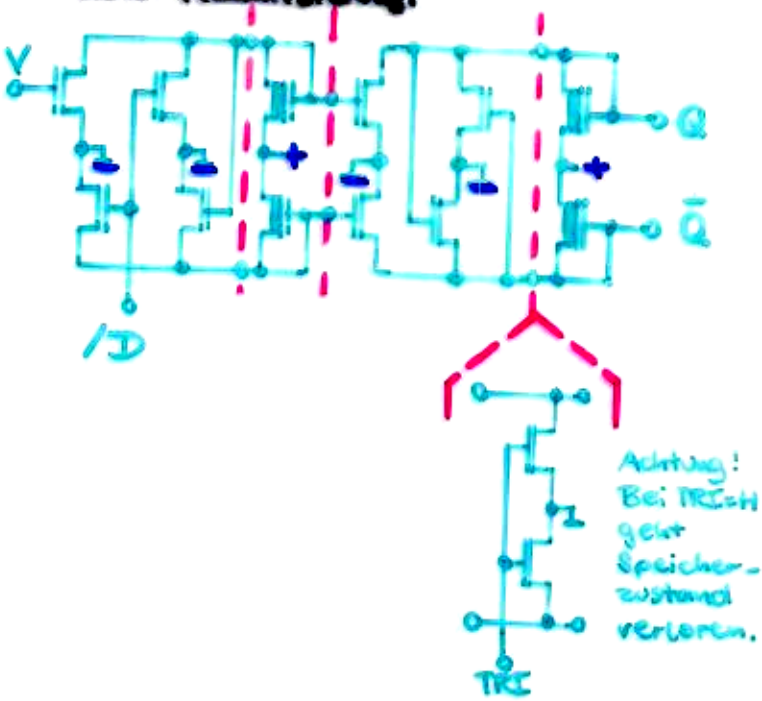




Stat. DV-Latch (aus RS-FF):



NOR-Planarisierung:

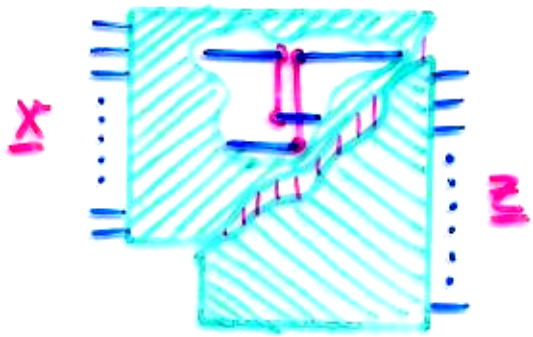


PLA-VARIANTEN:

DREIECKSUMFORMUNG:

- Vertauschung der Eingänge $x_1 \dots x_n$
- Vertauschung der Zwischenterme $y_1 \dots y_m$
- Vertauschung der Ausgänge $z_1 \dots z_i$

Ziel: Flächenreduzierung durch Dreiecksumformung:



Nachteil: Irrtumsmöglichkeit!

DECODIERTES EINGANGSFELD (AUSGANGSFELD)

Ziel: Schnelleres Layout / bessere Dreiecksumformbarkeit bei gehäuften Doppelterm ausdrücken

wie: $y_n = x_2 \bar{x}_3 \dots \vee \bar{x}_2 x_3 \dots \vee x_2 x_3 \dots$