DIGITALTECHNIK

Transistoren

nicht behandelt!

Kombinatorische Schaltungen

Grundkonzept

Eingänge $\underline{x}$ Ausgänge $\underline{y}$

$\underline{y} = f(\underline{x})$

Eigenschaften

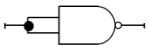
Keine Rückkopplung
Kein Speichervermögen

Ziel

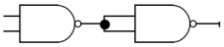
Ausgegl. Signalaufzeichnungen
Wenige, gleiche Bauelemente

NAND als …

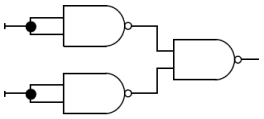
NOT



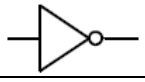
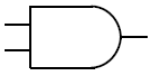
AND



OR



Bausteine

NOT (Inverter)	**AND**	**OR**	**XOR** (exkl. oder)	**NAND**	**NOR**
					
$\sim$	$*$	$+$	$-$		

x	y
0	1
1	0

x_0	x_1	y
0	0	0
0	1	0
1	0	0
1	1	1

x_0	x_1	y
0	0	0
0	1	1
1	0	1
1	1	1

x_0	x_1	y
0	0	0
0	1	1
1	0	1
1	1	0

x_0	x_1	y
0	0	1
0	1	1
1	0	1
1	1	0

x_0	x_1	y
0	0	1
0	1	0
1	0	0
1	1	0

$\sim(\sim x) = x$ $\sim x + x = 1$	$x * 0 = 0$ $x * 1 = x$	$x + 0 = x$ $x + 1 = 1$	y = Inputs mit 1 MOD 2	not and	not or

Regeln

Kommutativgesetz	**Assoziativgesetz**	**Distributivgesetz**	**Gesetz von de Morgan**
$x * y = y * x$ $x + y = y + x$ $x - y = y - x$	$(x * y) * z = x * (y * z)$ $(x + y) + z = x + (y + z)$ $(x - y) - z = x - (y - z)$	$(x + y) * z = (x * z) + (y * z)$ $(x * y) + z = (x + z) * (y + z)$	$\sim x * \sim y = \sim(x + y)$ $\sim x + \sim y = \sim(x * y)$

Vereinfachungen

$x * \sim y + \sim x * y = x - y$	$x + \sim x * y = x + y$	$(x + y) * (\sim x + z) = x * \sim x + x * z + y * \sim x + y * z$
$x * y + \sim x * y = y$	$x * (\sim x + y) = x * y$	$\sim x * \sim y + \sim x * y + x * \sim y + x * y = 1$

Wahrheitstabelle -> Schaltung

a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

1. Schritt
   Alle 1 nehmen und Schaltung in AND-Formen aufschreiben
2. Schritt
   DNF (Disjunktive Normalform) aufstellen
   $f(a,b,c) = \sim a * b * c + a * \sim b * \sim c + a * \sim b * c + a * b * \sim c + a * b * c$
3. Schritt
   Vereinfachung in Karnaugh-Map

Karnaugh-Map

	ab	$a\sim b$	$\sim a\sim b$	$\sim ab$
c	1	1	0	1
$\sim c$	1	1	0	0

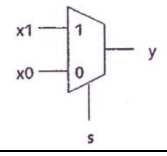
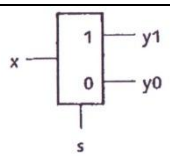
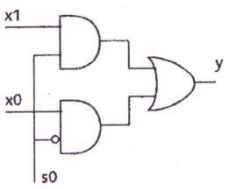
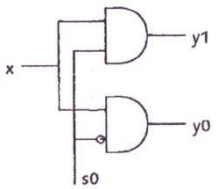
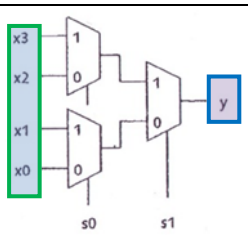
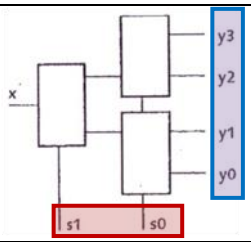
Alle nicht gegenteilige Komponenten aufschreiben
$f(a,b,c) = a + bc$

Regeln

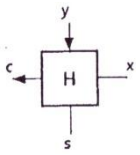
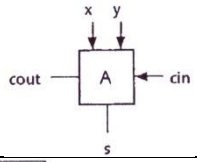
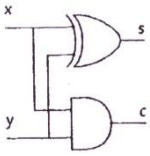
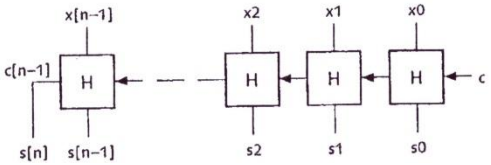
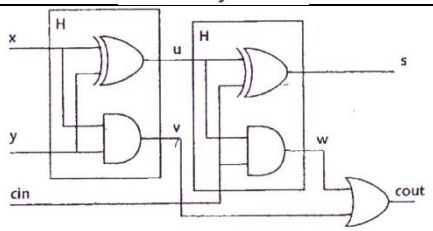
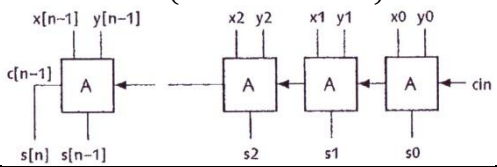
- Von Spalte zu Spalte nur ein Argument ändern
- Benachbarte Terme (2x1, 2x2, 4x1,4x2,4x2,…) zusammenfassen
- Periodisch fortlaufend vorstellen

	Binäre Darstellung	**Zweierkomplement**	
Darstellung	Nur positive Zahlen x_3 x_2 x_1 x_0 2^3 2^2 2^1 2^0	positive und negative Zahlen x_3 x_2 x_1 x_0 -2^3 2^2 2^1 2^0	**Negation** Alles umkehren, +1 rechnen **Addition** $0+0 \rightarrow 0$ $0+1 \rightarrow 1$ $1+1 \rightarrow 0\ (c=1)$ **Subtraktion** Addition mit negiertem Wert **Abschneidung = kein Fehler** **Overflow = Fehler in Rechnung** Wenn letzte 2 carry's (Übertrag) verschieden
Wertebereit	$W = [0..2^n - 1]$	$W = [-2^{n-1} \ldots + 2^{n-1} - 1]$	
Zahlen $a = msg$ (most significant bit)	a b c 10_{er} 0 0 0 0 0 0 1 1 0 1 0 2 0 1 1 3 1 0 0 4 1 0 1 5 1 1 0 6 1 1 1 7	a b c 10_{er} 1 0 0 −4 — $1.\ldots_2$ *negativ* 1 0 1 −3 1 1 0 −2 1 1 1 −1 0 0 0 0 — $0000_2 = 0$ 0 0 1 1 — $0.\ldots_2$ *positiv* 0 1 0 2 0 1 1 3	

Multiplexer

Multiplexer (2-1 mux) oder Selektoren	**Demultiplexer** (1-2) oder Decoder	**Kaskadierter Multiplexer**	**Kaskadierter Demultiplexer**
		$3 * (2-1\ mux)$ $4-1\ mux$	$3 * (1-2\ demux)$ $2-4\ demux$
			
$y = x_0 * {\sim}s + x_1 * s$	$y_1 = x * s_0$ $y_0 = x * {\sim}s_0$	$y = x_0 * {\sim}s_0 * {\sim}s_1$ $+x_1 * s_0 * {\sim}s_1$ $+x_2 * {\sim}s_0 * s_1$ $+x_3 * s_0 * s_1$	$y_3 = x * s_1 * s_0$ $y_2 = x * s_1 * {\sim}s_0$ $y_1 = x * {\sim}s_1 * s_0$ $y_0 = x * {\sim}s_1 * {\sim}s_0$

Addierer (addieren zwei 1-digit Zahlen)

Halbaddierer	**Inkrementer** (zählt +1)	**Volladdierer** (addiert drei binäre inputs)
	$n * Halbaddierer$ $Inkrementer$	
 c = Übertrag s = Summe		
x y c s 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 0	**Negation im 2er-Komplement** $Halbaddierer\ mit$ $negierten\ Eingängen$ $Negation$	**N-bit Addierer** $(n * Volladdierer)$ 

Sonstige

Addierer mit Fast Carry Generation

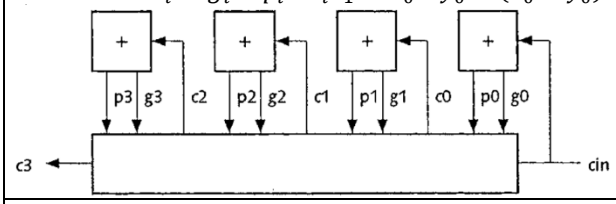
Übertrag

$c_0 = g_0 + p_0 * c_{in}$

$c_1 = g_1 + p_1 * g_0 + p_1 * p_0 * c_{in}$

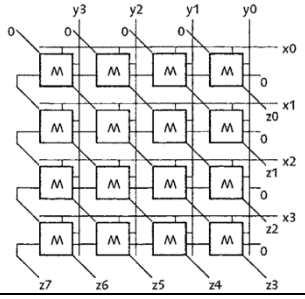
schneller Torlaufzeit (immer 3T)
benötigt mehr Platz

$$c_i = g_i + p_i * c_{i-1} = x_0 * y_0 + (x_0 - y_0) * c_{in}$$



The Multiplier

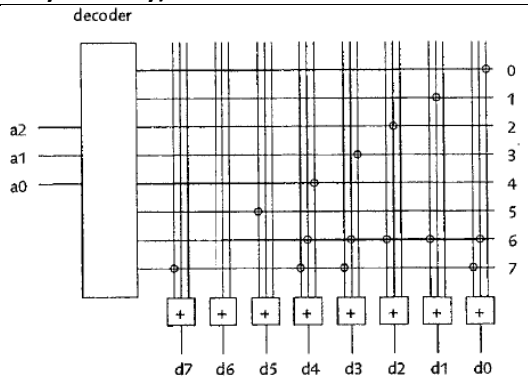
Multiplikation von $n * n$-bit



$(x * y = z)$

$x: n\ bit$
$y: n\ bit$
$z: n^2\ bit$

ROM (Read-Only Memory)



ROM-Typen

ROM
16*8 ROM → a=4, d=8
mit FET (Field Effect Transistor)

PROM
Thermisches zerstören von Transistoren

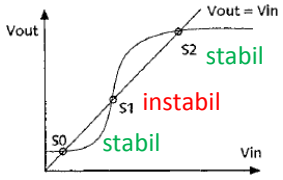
EPROM
Zerstören mit UV-Licht

E^2-PROM

FLASH-ROM
Sektoren-einteilung

Latches (« Fallen ») + Register

Speicherprinzip

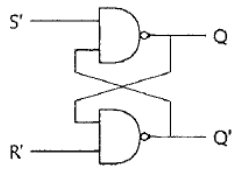


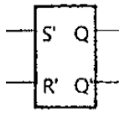
2 Betriebszustände
(S0, S2) -> 1 Bit

Q‘ = Nullauslöser

Bausteine

SR-Latch (Set-Reset)

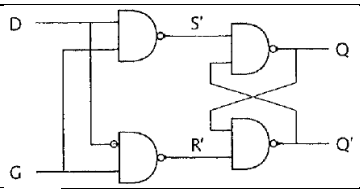


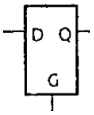


S‘ Set
R‘ Reset
Q Output

R'	S'	Q^+	
1	1	Q	storing
1	0	1	set
0	1	0	reset
0	0	1	

D-Latch
gibt wenn G=1 das D aus

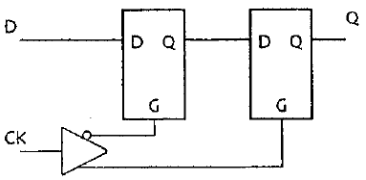


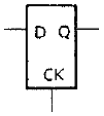


D Input
G Gate
Q Output

G	D	Q^+	
0	x	Q	storing
1	0	0	transparent
1	1	1	Q=D

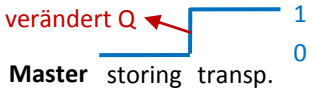
D-Register (D-Flip-Flop)
gibt all Takt das D aus (wenn G an CK ist)





D Input
CK Clock
Q Output

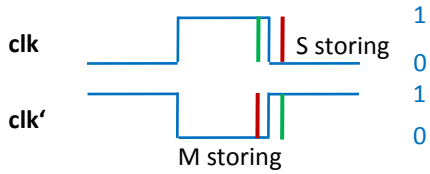
2 D-Latch -> immer einer im storing:



Master storing transp.
Slave transp. storing

Synchrone, Sequentielle Schaltungen

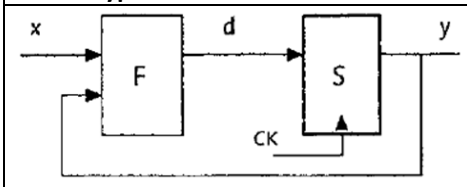
Clock



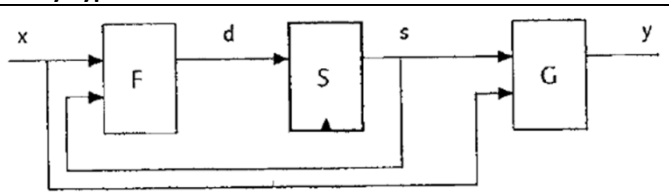
unproblematisch
problematisch

The State Machine

Moore-type state machine

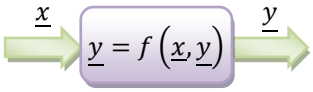


Mealy-type state machine

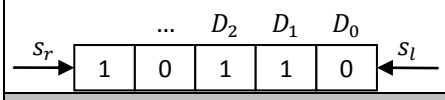


Bausteine

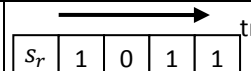
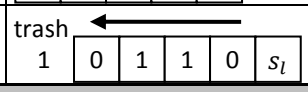
F = next state function (n.s.f)	Kombinatorische Schaltung
S = state (s.)	Auswahl an Registern (n-Register = 2^n-states) Die alle das gleiche Clock-Signal verwenden
G = output function (o.f.)	Zweite Kombinatorische Schaltung
reset	Um Schaltung in Initial-Zustand zu bringen
One-hot state machine	Jeder state wird von einem Register gehalten

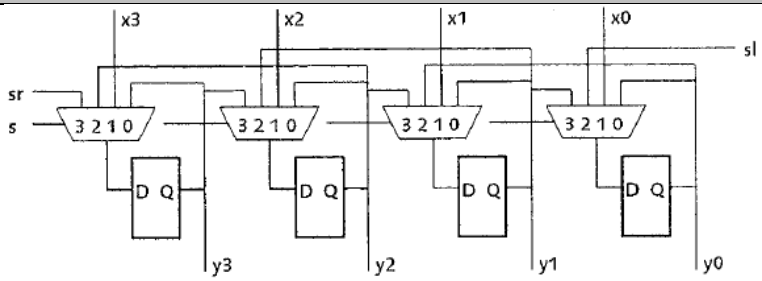


Schieberegister (shift register)



Modi

Hold	Inhalt behalten
load	D-> Schieberegister
shift right	
shift left	

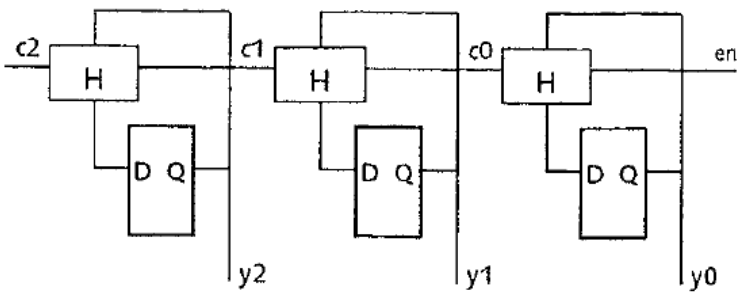


mithilfe Multiplexer

Counter

Variationen

BCD	4 Bit	0..9
Binär	4 Bit	0000..IIII



mithilfe Inkrementer

Schrittmotor