7 Logische Netze

In diesem Kapitel wollen wir uns zunächst den Grundprinzipien der Realisierung von Logik-Schaltungen zuwenden. Wir betrachten die Logik-Modellierung durch einfache ***Schalternetze***, wobei nur ideale Schalter verwendet werden. (Diese Technologie-unabhängige Abstraktionsebene wird gern als *Switching-Ebene (switching level)* bezeichnet. Solche *Schalternetze* enthalten nur abstrakte Schalter und sind deshalb nicht mit *Schaltnetzen* zu verwechseln (nicht speichernde Gatternetze).) Darauf aufbauend wird die Modellierung von Potential-Logik behandelt. In einem späteren Kapitel (8) wird gezeigt, wie eine Hardware-Beschreibungssprache (KARL-3 [2]) eingesetzt werden kann, um durch *Pseudo-Switching-Level*-Beschreibungen Bus-Systeme und andere Schaltungen in der Switching-Ebene zu modellieren und zu simulieren.

7.0 Verwendete Symbole und Formelzeichen

Im Gegensatz zur Mathematik und zur Elektrotechnik, die bei formalen Notationen spezielle Zeichen (wie $\int, \Sigma, \&, \neg, ...$) bevorzugen, ziehen wir in der Informatik Namen vor (beispielsweise *and, or, not,...*), nicht zuletzt wegen der damit erzielbaren besseren Verständlichkeit komplexer Strukturen. Eine weitere Motivation, um einen überquellenden Vorrat spezieller Symbole zu vermeiden, ist die Unterstützung der Computer-Anwendung durch einen genormten maschinenlesbaren Zeichenvorrat, wie z.B. den ASCII-Zeichensatz. Diese Präferenz gilt nicht nur für Operatoren und Funktionen etc., sondern auch für Variablen, wie z.B. Signalnamen: statt n1, n2, n3,... bevorzugen wir ADDEND, AUGEND, SUMME, IN_CARRY, OUT_CARRY etc. Wir suchen eine Vermeidung des Analphabeten-Syndroms nicht nur für rein textuelle Darstellungen, sondern auch im Rahmen von Diagrammen und anderen graphischen Notationen. Bild 7.2 veranschaulicht diese Philosophie.

7.1 Schaltalgebra - auch außerhalb der Logik-Ebene

Logische Netze wie die in diesem Kapitel behandelten Transmissionsnetze sind eng mit der Schaltalgebra (Boole'sche Algebra) verflochten. Die Anwendung sukzessiver algebraischer

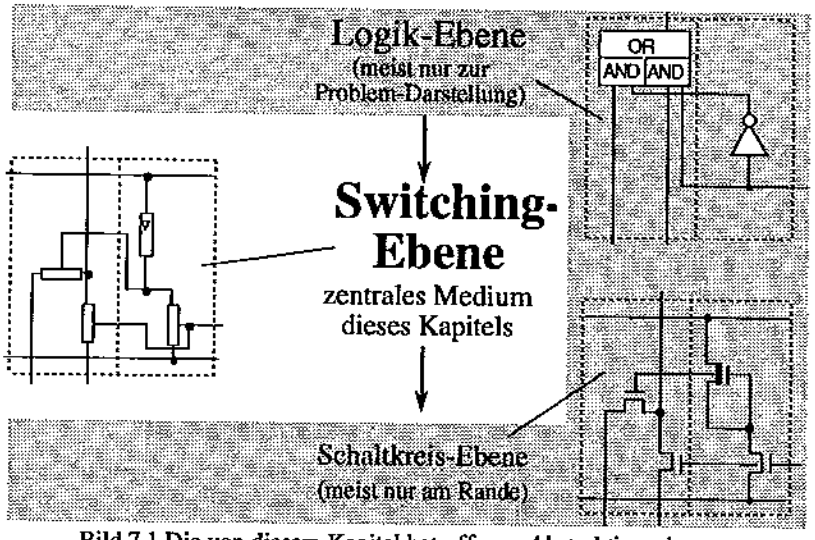


Bild 7.1 Die von diesem Kapitel betroffenen Abstraktionsebenen

Äquivalenz-Transformation ist auch eine Möglichkeit heuristischen Vorgehens u. a. beim strukturierten VLSI-Entwurf (vgl. Abschnitt 20.4.2).

Zunächst werden Regeln der Schaltalgebra vorgestellt, die den heuristischen Entwurfsprozeß unterstützen und helfen können, die Anzahl der Gatter zu senken. Diese Regeln, die über die Netzumwandlung auf Gatterebene veranschaulicht werden (Bild 7.3), sind überwiegend einfach anzuwenden und werden in der digitalen Logik gelehrt. Als Beispiele hierfür seien die *De Morgan'sche Regel* (Bild 7.3 g) genannt und die Tautologie: Doppelte Negation von A ergibt A (Bild 7.3 f). Etwas umfangreicher, aber nicht zu kompliziert ist das *Shannon'sche Theorem* (Bild 7.3 j) sowie das *doppelte Shannon'sche Theorem* (Bild 7.3 k). Am besten kann man diese Umformungen mit Hilfe eines später verwendeten (Abschnitt 7.4), hier vorweggenommenen Beispiels verdeutlichen. Ein Transmissionsnetz (Bild 7.8 a) wird in ein Logik-Diagramm umgewandelt (Bild 7.9), das nun eine Redundanz zeigt, die im T-Netz nicht offensichtlich war. Dieses Logikdiagramm wird nun in ein KV-Diagramm überführt (Bild 7.10) und nach dessen Minimierung zurückverwandelt über ein Logikdiagramm (Bild 7.11) in ein T-Netz (Bild 7.12).

7.2 Entwurf logischer Gatter

Ein wichtiges Gebiet der Technischen Informatik sind die Prinzipien der Realisierung von logischen Gattern. Es gibt hierzu viele Möglichkeiten ("Technologien"): mechanische Logik, hydraulische Logik, pneumatische Logik, optische Logik, elektrische Logik und andere Realisierungsformen. Die Elektrizität ist also prinzipiell nicht notwendig für die Realisierung

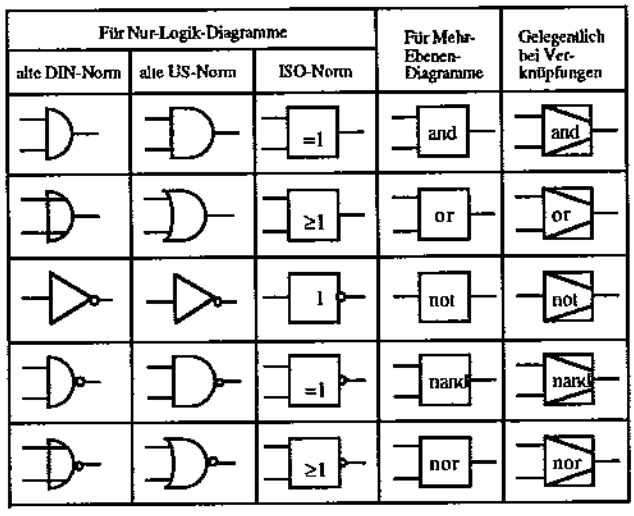


Bild 7.2: Verschiedene Logik-Diagramm-Notationen.

von logischen Netzwerken. Dies ist ein Argument dafür, daß Schaltnetz- und Schaltwerktheorie zur Informatik gehört und nicht zur Elektrotechnik. Die elektrische Realisierung ermöglicht jedoch die höchsten Schaltgeschwindigkeiten, die höchste Packungsdichte und die rationellste Massenherstellung. Letztlich behandeln wir hier deshalb elektrische Logik-Realisierungen. Wir führen zunächst eine technologieunabhängige Modellierung ein. Diese soll uns dabei helfen, allgemeine Realisierungs-Prinzipien zu identifizieren, die sich dann auch für alle üblichen elektrischen Technologien anwenden lassen, z.B. für die Anwendung von Relais, Dioden, bipolaren Transistoren, MOS-Transistoren, etc. Wir benützen dabei drei Abstraktionsebenen (vgl. Bild 7.4):

- die Logik-Ebene (engl.: *gate level*, die Ebene der Problemformulierung),
- die CSA-Ebene (engl.: *switching level*, die Ebene der Modellierung), sowie
- die Schaltkreis-Ebene (engl.: *circuit level*, Ebene der technischen Realisierung).

CSA steht hierbei für *Connector/Switch/Attenuator* (Verbinder/Schalter/Abschwächer). Diese Bezeichnungen werden später erklärt. Wenn auch noch Kapazitäten (in dieser Ebene "well" w genannt) noch mit einbezogen werden, dann wird diese Ebene auch CSAW-Ebene genannt.

In der Logik-Ebene gibt es nur gerichtete Elemente (Gatter), an denen Eingänge und der Ausgang streng voneinander unterschieden werden (vgl. Bild 7.11). Wenn Ein- und Ausgänge einmal nicht durch Pfeile (vgl. Bild 7.9) gekennzeichnet sind, so lassen sich diese leicht aus dem Kontext heraus identifizieren (vgl. Bild 7.11). In der Switching-Ebene (switching level) hingegen werden ungerichtete Elemente verwendet bei einem Schalter. Im Gegensatz zu An-

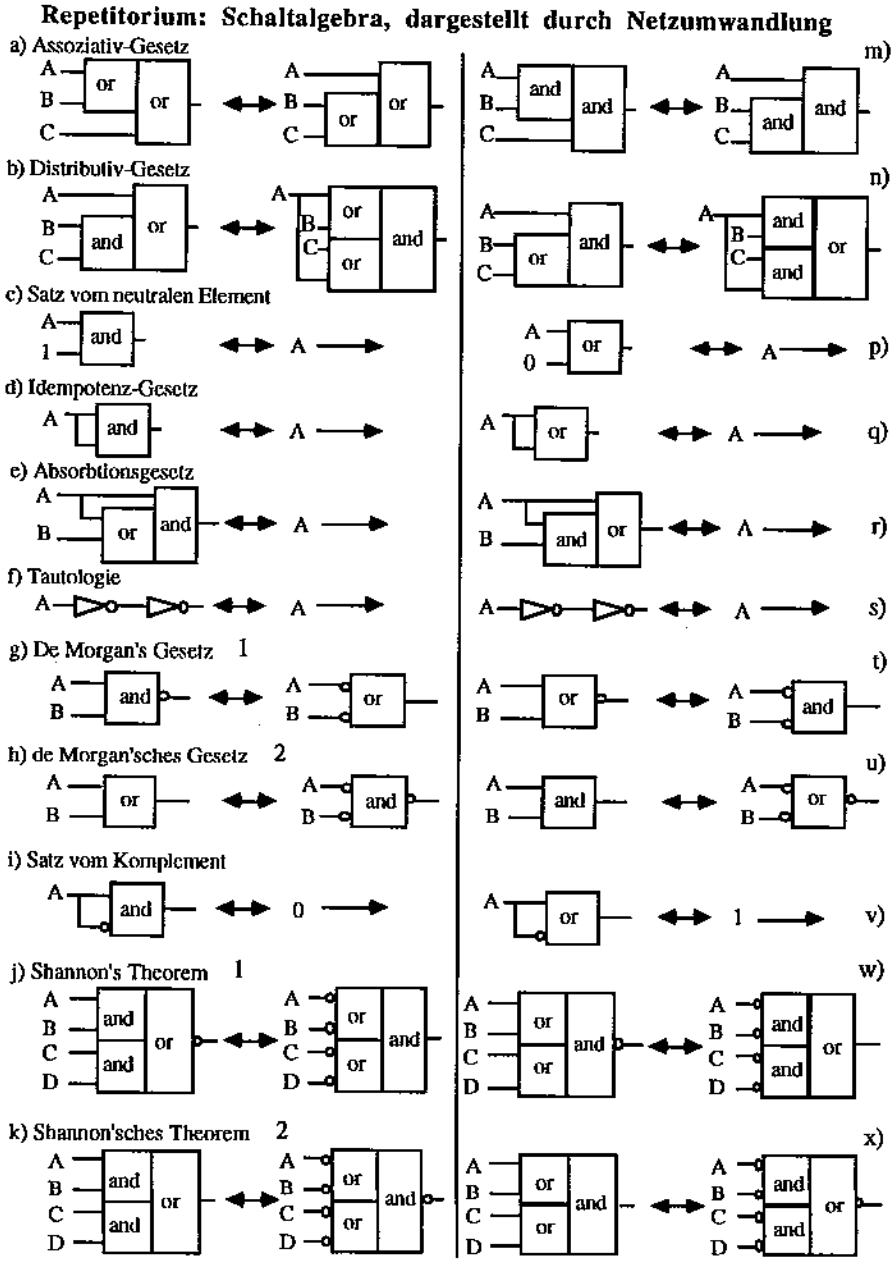


Bild 7.3: Regeln der Schaltalgebra, ausgedrückt in Logik-Diagrammen.

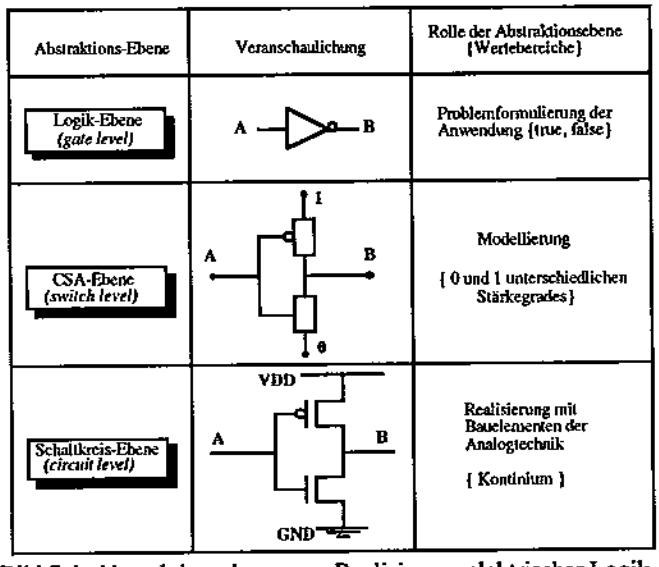


Bild 7.4: Abstraktionsebenen zur Realisierung elektrischer Logik.

schlußpins von Gattern sind die Knoten von Elementen der Switching-Ebene bezüglich der Eigenschaft als Eingang oder Ausgang nicht festgelegt. Bei einem Schalter, beispielsweise ist die Richtung des Stromflußes oder Informationsflusses für den Schalter nicht festgelegt, sondern ergibt sich erst durch Ausgleichsvorgänge unter dem Einfluß anderer angeschlossener Elemente aus der Schaltungsumgebung. In der Logik-Ebene haben wir für ein Bit die aus der Digitalen Logik bekannten Bit-Werte: $V_L=\{0,1,*,?\}$ mit der Bedeutung:

0 = 'false' (falsch)

1 = 'true' (wahr)

Hinzu kommen in der Theorie noch zwei Pseudo-Werte:

* = 'don't care' (beliebig: 0 oder 1)

? = 'unknown' (unbekannt)

Manchmal findet man in der Literatur für das Zeichen '*' die Symbole 'd' ('don't care') oder 'z' ('high impedance' oder 'hochohmig') und für das Zeichen '?' ein 'U' ('undefiniert' oder 'undefined') oder 'x' für 'unbekannt'. Geht man nun in die Switch-Ebene hinunter, so erhöht sich im allgemeinen der Werte-Vorrat. Die Wahrheitswerte '0' oder '1' können eine unterschiedliche logische Stärke ("logical strength") haben, wie z.B. '0' für 'starke 0' oder '~0' für 'schwache 0', und '~~0' für 'noch schwächere' Nullen. Die "Stärken-Ordnung" eines solchen Werte-Vorrates wird

durch ein sogenanntes Hasse-Diagramm definiert (Beispiel in Bild 8.4 a). In der Switching-Ebene können wir 2 Arten von Schaltungen unterscheiden:

- reine Schalternetze
- CSA-Netzwerke oder CSAW-Netzwerke

Reine Schalternetze können ohne Verwendung "weicher" Logikvariablen auf ungerichtete Weise charakterisiert werden durch "Durchlaßfunktionen" mit den Wahrheitswerten '0' (undurchlässig) und '1' (durchlässig). Solche reinen CS-Netze eignen sich gut für die direkte Modellierung von Logik mit Transfer-Transistoren und auch für CMOS-Schaltungen. Erst bei CSA-Netzwerken, die erst in Kapitel 8 behandelt werden, werden "weiche" Wahrheitswerte benützt, da neben Schaltern auch Widerstände (und in CSAW-Netzwerken auch Kondensatoren) modelliert werden müssen. Die einfachste Form eines Schalternetzes ist der logische Zweipol, der durch eine einzige Durchlaßfunktion charakterisiert ist.

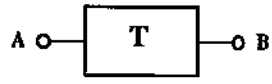


Bild 7.5: logischer Zweipol.

7.3 Logik-Modellierung durch einfache Schalternetze

Zunächst sei nun noch der Begriff des logischen Zweipols erklärt (Bild 7.5). Logische Zweipole sehen folgendermaßen aus: sie bestehen aus einem Transmissionsnetz T mit je einem Anschluß A und B. Wir können also diesen logischen Zweipole durch eine Transmissionsfunktion T charakterisieren, die den Wert T = "wahr" hat, wenn das Netz durchlässig ist, und wenn T den Wert "falsch" oder 0 hat, dann ist das Netz undurchlässig. Wir werden also nun versuchen durch eine Analyse herauszubekommen, was für eine reine logische Funktion ein solcher logischer Zweipol darstellt oder realisiert.

Logik-Typ	Zustand	logischer Wert
Transmissions-Logik	leitend ('on')	1
	undurchlässig ('off')	0
Potential-Logik	Potential hoch (high, 'hi' oder 'high')	1
	Potential niedrig (low, 'lo' oder 'low')	0

Bild 7.6: Vergleich zwischen Potential- und Transmissions-Logik

Zunächst werden die Begriffe *Transmissions-Logik* (z.B. Stromlogik) und *Potential-Logik* (z.B. Spannungslogik) erläutert, wobei wir uns anfangs der Einfachheit wegen auf CS-Netze beschränken wollen, d.h. Abschwächer werden noch nicht mitverwendet. Bei Stromlogik wird ein logischer Wert als Transmissionsfunktion dadurch definiert, daß wir sagen: wenn Strom fließen kann, liegt eine logische 1 vor, wenn kein Strom fließen kann, liegt eine log. 0 vor. Es gibt aber auch Potentiallogik (oder Verhältnislogik: dies wird später erklärt), etwa gemäß folgender Definition: ist das Potential (z.B. die elektrische Spannung) hoch, dann ist der logische Wert eine 1, ist das Potential niedrig, dann ist er 0. Wir fassen dies in Bild 7.6 zusammen.

Wir beginnen mit der Behandlung der Transmissions-Logik, weil diese eine Grundlage der Potentiallogik ist. Zuvor sei noch ein Teil der abstrakten Bauelemente der CSA-Ebene eingeführt (Bild 7.7). Die hier als Bauelemente verwendeten Schalter (switch) lassen sich leicht über elektromechanische Relais veranschaulichen. Der positive Schalter, ähnlich einem Relais mit einem Arbeitskontakt ('make contact' relais) ist durchlässig, wenn er aktiviert ist, d.h.

Steuereingang K = 1, jedoch undurchlässig im passiven Zustand, mit K = 0. Ein negativer Schalter, ähnlich einem Relais mit einem Ruhekontakt ('disconnecting' relais) ist undurchlässig, wenn er aktiviert ist (K = 1), jedoch leitend im passiven Zustand (K = 0).

7.4 Transmissions-Logik

Durch die Betrachtung von Strukturen der Transmissions-Logik erlernen wir die schaltungstechnischen Grundlagen für die Realisierung von Logikgattern. Durch *Analyse* einer MOS-Schaltung ermitteln wir die logische Funktion, damit man leichter einsehen kann, wie der umgekehrte Prozeß (die *Synthese*) funktioniert. Wir beschäftigen uns also zunächst einmal damit, wie man die Transmissionsfunktion T eines solchen Kontaktnetzwerkes findet, wobei

T = 1 (wahr), wenn das Netz durchlässig ist (etwa für elektrischen Strom),

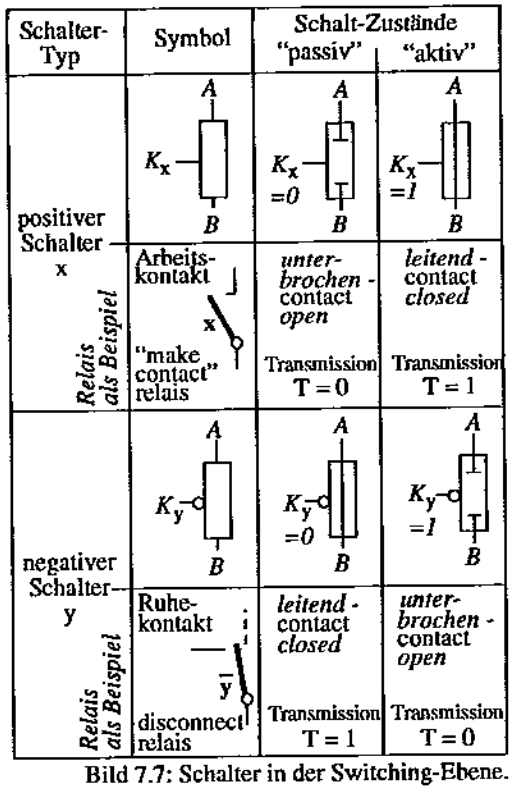


Bild 7.7: Schalter in der Switching-Ebene.

und

T = 0 (unwahr), wenn das Netz gesperrt, also undurchlässig ist.

In Bild 7.8 bis Bild 7.12 ist eine solche Analyse vereinfacht dargestellt. Aus dem Transmissions-Netzwerk (CS-Modell in Bild 7.8 b: der Begriff der *Verbindungsmenge* wird in Abschnitt 7.6 eingeführt), das beispielsweise durch Relais-Kontakte realisiert sein könnte (Bild 7.8 a), gewinnen wir zunächst ein Logik-Diagramm (Bild 7.9). Ebensogut ist auch eine äquivalente textuelle Darstellung möglich durch ein System von Boole'schen Funktionen. Deren Ermittlung soll jedoch erst später als Beispiel im Zuge der Veranschaulichung einer Serien/Parallel-Analyse erfolgen. Die Transmission T des Netzes sei über die Durchlässigkeit von A nach B definiert, wobei A der Eingang und B der Ausgang ist. Für jeden Durchlaß-Pfad (Bild 7.8 c) erhalten wir ein AND-Gatter. Die gefundene Funktion kann man durch Minimierung (Bild 7.10) vereinfachen und erhält ein neues Logikdiagramm (Bild 7.11).

Die Anzahl der Gatter wurde von 5 auf 3 reduziert. Hieraus kann man die Realisierung auf der Switching-Ebene in Bild 7.12 gewinnen. Die Existenz des Pfades Nr. 4 stört nicht, da dieser stets die Transmission null hat. In den folgenden Abschnitten wird dann näher auf die Analy-

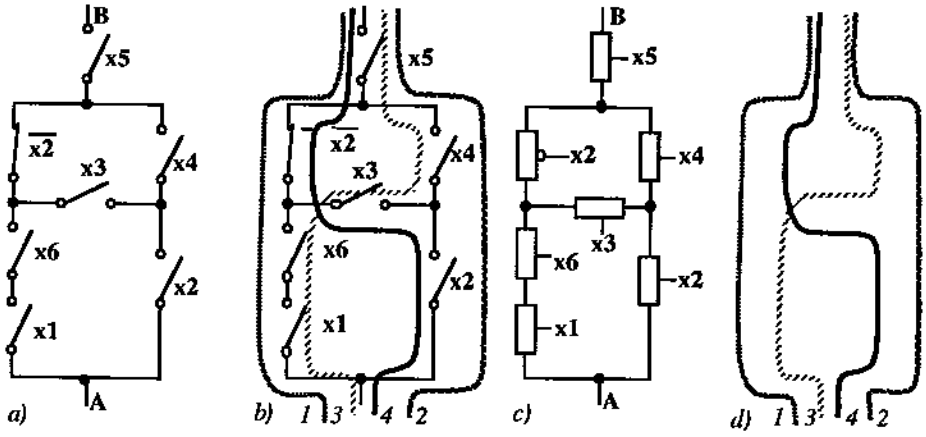


Bild 7.8: Transmissionsnetzwerk-Beispiel, a) Relaisnetz-Veranschaulichung, b) Relaisnetz mit Verbindungsmenge, c) CS-Modell, d) alle Pfade durch das Netz (Verbindungsmenge).

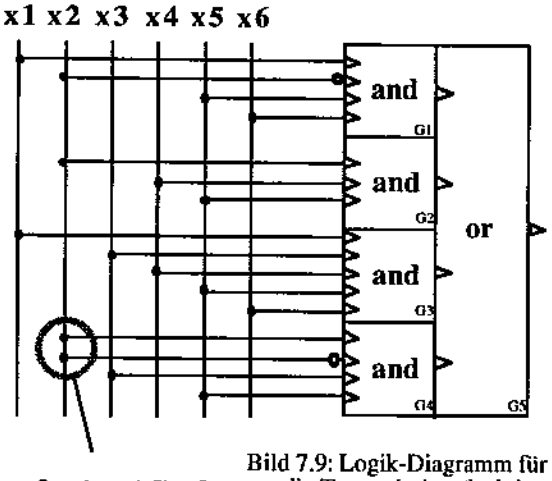


Bild 7.9: Logik-Diagramm für die Transmissionsfunktion des Netzwerkes aus Bild 7.8

severfahren eingegangen. Es werden Verfahren vorgestellt, mit deren Hilfe man die logische Funktion auch von komplexeren Transmissionsnetzwerken ermitteln kann.

Je nach anwendbaren Analyseverfahren gibt es zwei Klassen von Transmissions-Netzwerken: Die erstere Klasse ist die der sogenannten *Serien-Parallel-Netzwerke*, die man durch eine Serien/Parallel-Zerlegung analysieren kann (über eine sukzessive Zerlegung in zueinander parallele oder in Serie liegende Subnetzwerke). Es gibt aber auch die letztere Klasse, bei welcher man das Analyseproblem auf diese Weise nicht lösen kann. Diese Netze werden dann *Nicht-Serien-Parallel-zerlegbare-Netzwerke* genannt.

7.5 Serien/Parallel-zerlegbare Schalternetze

Am Beispiel aus Bild 7.13 wird nun die Analyse von Serien/Parallel-Netzwerken dargestellt. Hierbei wird das Netz durch sukzessive Durchführung von Serien-Zerlegungen und Parallel-

Zerlegungen partitioniert, solange bis nur noch atomare Teile übrig bleiben, d.h. Teile, die maximal nur noch einen einzigen Schalter enthalten.

Es wird dabei nach folgenden Regeln vorgegangen: liegt eine Serienschaltung vor, d.h sind Teilnetze bezüglich des Stromflusses in Serie geschaltet, so wird eine AND-Verknüpfung notiert. Liegt hingegen eine Parallelschaltung vor, d.h. die Pfade verlaufen nebeneinander, so entspricht das einer logischen OR-Verknüpfung (vgl. Bild 7.14).

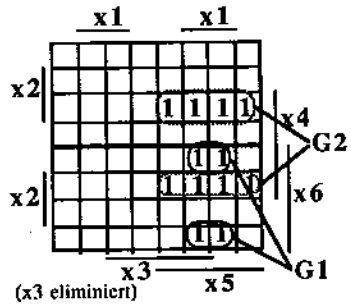


Bild 7.10: KV-Diagramm zur Minimierung der Funktion aus Bild 7.9.

Wir wollen nun am Beispiel in Bild 7.13 demonstrieren, wie eine Serien/Parallel-Analyse durchgeführt wird. In Bild 7.15 ist die Schaltung in ihre Subnetze zerlegt. Im einzelnen wurden dabei folgende Schritte durchgeführt: Zunächst läßt sich das gesamte Transmissionsnetz T in die beiden Subnetze T1 und x5 zerlegen (Serienschaltung). Also:

$$T = T1 \text{ and } x5 \qquad (7.1)$$

Danach läßt sich T1 in die Netze T2 und T3 aufspalten (ebenfalls Serienschaltung).

$$T1 = T2 \text{ and } T3 \qquad (7.2)$$

Damit ergibt sich für (1): T = T2 and T3 and x5. T2 zerfällt in T4 und x2, T3 in x2 und x4 (jeweils Parallelschaltung).

$$T2 = T4 \text{ or } x2 \qquad (7.3)$$

$$T3 = \text{not}(x2) \text{ or } x4 \qquad (7.4)$$

Daraus ergibt sich für (1): T = (T4 or x2) and (not(x2) or x4) and x5.T4 spaltet sich in x1 und x6 auf (Serienschaltung).

$$T4 = x1 \text{ and } x6 \qquad (7.5)$$

Somit sind nun alle Subnetze atomar und es ergibt sich hieraus die folgende Boole'sche Gleichung für T:

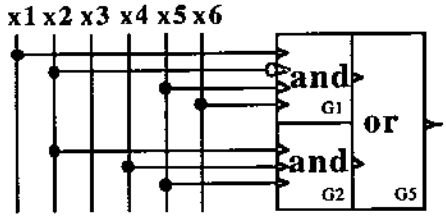


Bild 7.11: Logik-Diagramm für die minimierte Transmissionsfunktion zu Bild 7.9

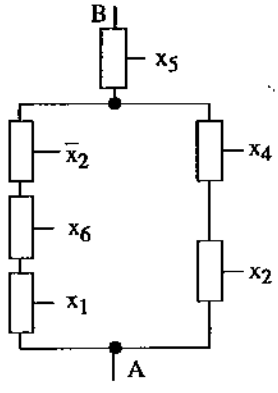


Bild 7.12: Realisierung der (rück-)gewonnenen Funktion der Switching-Ebene.

$$T = [\{(x1 \text{ and } x6) \text{ or } x2\} \text{ and } (\text{not}(x2) \text{ or } x4)] \text{ and } x5. \qquad (7.6)$$

Somit haben wir aus der Struktur des Schalternetzes (T-Netz) in Bild 7.13 die Transmissions-Funktion T als eine Verhaltensbeschreibung gewonnen.

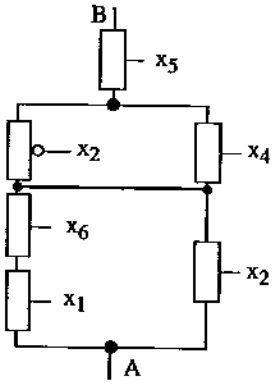


Bild 7.13: T-Netzwerk.

7.5.1 Definitionen und Beispiele

Def.: Die Menge der Transmissionsausdrücke (T-Ausdrücke) über einer endlichen Menge $\{x_1 .. x_n\}$ von Grundsymbolen (Eingangsleitungen) wird rekursiv definiert durch:

i. x_i (auch $\bar{x}_i$) ist ein T-Ausdruck für $1 \le i \le n$ (atomarer T-Ausdruck)

ii. Wenn e_1, e_2 T-Ausdrücke sind, die kein gemeinsames Grundsymbol enthalten, dann sind auch $(e_1 \wedge e_2)$ und $(e_1 \vee e_2)$ T-Ausdrücke

iii. T-Ausdrücke können nur gemäß i. u. ii. gebildet werden

Beispiele: Seien a, b, c die Eingänge. Dann sind a, b, c, $(a \wedge b)$, $(a \wedge c)$, $((a \vee b) \vee c)$, und $(a \wedge (b \vee c))$ Beispiele für T-Ausdrücke (vgl. Bild 7.17).

Bem.: Die Menge der T-Ausdrücke für eine gegebene Menge von Eingangsleitungen ist endlich.

Definition: Ein T-Netz T ist ein 5-Tupel $T = (N, E, f: E \rightarrow \{P \subset 2^N : |P| = 2\}, A, B)$ wobei:

N endliche Mengen von Knoten

E endliche Mengen von Transmissionsausdrücken

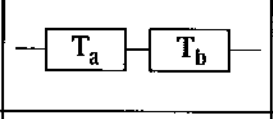
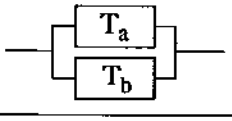
T-Netzwerk		
Boole'scher Ausdruck (T-Ausdruck)	T_a and T_b	T_a or T_b

Bild 7.14: Abbildung zwischen T-Netzwerk und Boole'schem Ausdruck (T-Ausdruck)

f gesamte Transmissionsfunktion (T-Funktion)

A, B ∈ N ausgezeichnete Knoten (Anfangs- und Endknoten)

Anfangs- und End-Knoten sind in Transmissions-Diagrammen repräsentiert durch einen Kreis-Ring, während alle anderen Knoten durch eine schwarze Kreisfläche dargestellt sind (vgl. Bild 7.17).

Beispiel: Das Transmissionsnetz

T = ({A, B, C}, {x1, x2, x3, x4}, f, A, B),

wobei f festgelegt ist durch

f(x1) = {A,B}, f(x2) = {C,A},

f(x3) = {C,B} und f(x4) = {C,B}

läßt sich graphisch darstellen wie in Bild 7.18.

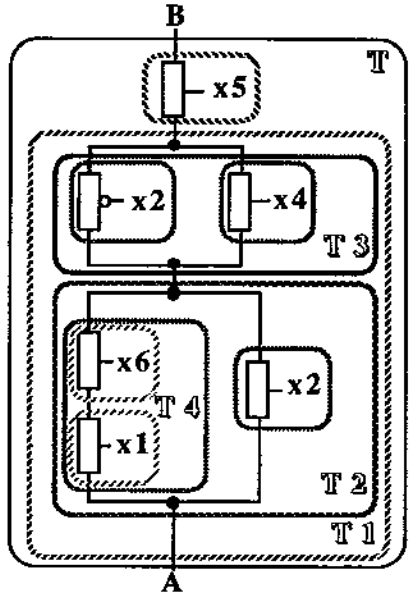


Bild 7.15: Serien/Parallel-Zerlegung des Transmissionsnetzes aus Bild 7.13.

Bemerkung: Die Kanten entsprechen Schaltern, die jeweils mit einer Eingangsleitung (schattiert) markiert sind. f ordnet jeder Kante die durch sie verbundenen beiden Knoten zu. Beispielsweise f = {J, K} besagt, daß f die Kante zwischen J und K ist.

A, B	ausgezeichnete Knoten	K	Knotenfolge
A	Anfangsknoten	L_n	Layout der nMOS-Seite
B	Endknoten	L_p	Layout der pMOS-Seite
e, e_i	Transmissions-Ausdruck	M	Menge von Folgen aus Elementen von E, wobei $M = \{F_1, \ldots F_m\}$
E	endliche Menge $\{\ldots e_i \ldots\}$ von Transmissionsausdrücken		
$f(e_i)$	e_i zugeordnete Kante	N	Kantenfolge
F	Kantenfolge, wobei $F_l = [e_{l1}, \ldots e_{lkl}]$, und $F_m = [e_{m1}, \ldots e_{mkm}]$	p, p_i	Kante
		T	Transfernetz T = (N, E, f, A, B)
		X	Menge v.Transfer-Ausdrücken

Bild 7.16: In Unterkapitel 7.5 und 7.6 verwendte Symbole und Formelzeichen

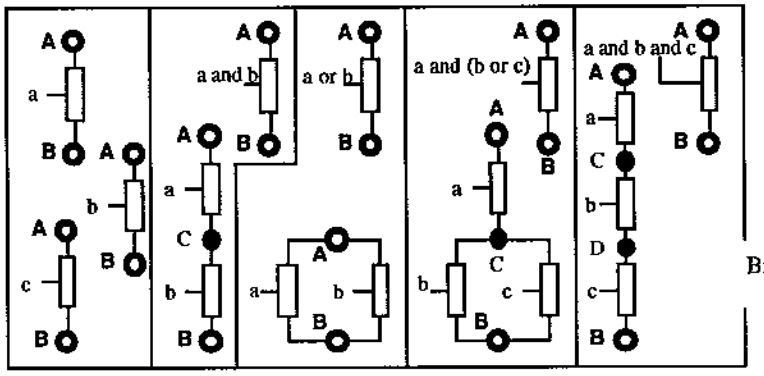
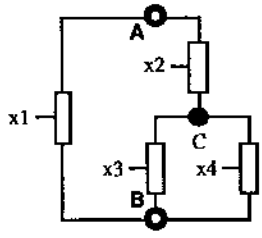


Bild 7.17: Atomare T-Ausdrücke

7.5.2 Ermittlung des T-Ausdruckes aus einem Transmissionsnetz

Dieser Abschnitt führt einen Algorithmus zur Serien/Parallelen-Zerlegung eines Transmissionsnetzes und zur Ermittlung des zugeordneten Transmissionsausdruckes ein. Er dient der Ermittlung eines T-Ausdruckes (Transmissionsausdrucks) e bei gegebenem T-Netz (Transmissions-netz) T_0.



e	f(e)
x1	{A, B}
x2	{C, A}
x3	{C, B}
x4	{C, B}

Bild 7.18: Transmissionsgraph; a) graphische Darstellung, b) textuelle Notation.

Eingabe: Transmissionsnetz
$T_0 = (N, E, f, A, B)$

Ausgabe: Fehlermeldung, falls T_0 nicht serien-parallel zerlegbar, ansonsten den T_0 zugeordneten Transmissionsausdruck e_{T0}

Verfahren: begin
$T := T_0$;
repeat

$T_{old} := T$;
sei $T = (N, E, f, A, B)$
if $\exists\, e_1, e_2 \in E: f(e_1) = f(e_2)$ *(gemeinsame Endknoten)*
then
$T := (N, E - \{e_1, e_2\} + \{(e_1 \vee e_2)\}, f', A, B)$, wobei
$f'(e) = f(e)$ falls $e \notin \{e_1, e_2\}$ und
$f'((e_1 \vee e_2)) = f(e_1)$;

$$\text{if } \exists\, X \in N - \{A, B\},\ e_1, e_2 \in E : X \in f(e_1) \cup f(e_2) \quad \wedge$$
$$|f(e_1) \cup f(e_2)| = 3 \quad \wedge$$
$$X \notin \bigcup_{e \in E - \{e_1, e_2\}} f(e) \qquad (7.7)$$

then

$T := (N - \{X\}, E - \{e_1, e_2\} + \{(e_1 \wedge e_2)\}, f', A, B)$, wobei

$f'(e) = f(e)$ falls $e \notin \{e_1, e_2\}$ und

$f'((e_1 \wedge e_2)) = f(e_1) \cup f(e_2) - \{X\}$

until $T = T_{old}$;

if $|E| = 1$ then write (E)

else write ('e_{T0} nicht serien-parallel-zerlegbar')

end.

Erläuterung. Hier werden sukzessive einfachere Transmissionsnetze erzeugt, indem einfache Parallelschaltungen bzw. Serienschaltungen von zwei mit Ausdrücken e_1 und e_2 markierten Zweigen durch einen einzigen mit dem Ausdruck $(e_1 \vee e_2)$ bzw. $(e_1 \wedge e_2)$ markierten Zweig ersetzt werden.

Dies wird solange wiederholt, bis das Netz keine solchen einfachen Schaltungen mehr enthält. Besteht zum Schluß der Graph nur noch aus einer einzigen Kante zwischen A und B, so ist die Zerlegung gelungen und die Kante mit dem gesuchten Ausdruck markiert.

Beispiel: Für das Transmissionsnetz aus Abschnitt 7.5.1 . liefert der Algorithmus als Ausgabe den Transmissionsausdruck: $(x1 \vee (x2 \wedge (x3 \vee x4)))$

Der obige Spezifikations-Formalismus sei an Hand von Bild 7.19 veranschaulicht (Beispiel nach Bild 7.18) und erklärt für diejenigen, die mit dieser Notation nicht gut vertraut sind. Zunächst werden in der Transmissionstabelle (Bild 7.19 b) zwei Zeilen 1 und 2, bzw. zwei Ausdrücke e_1 und e_2 mit übereinstimmenden Knotenmengen $f(e_1)$ und $f(e_2)$ gesucht:

$$\text{if } \exists\, e_1, e_2 \in E: f(e_1) = f(e_2)$$

wird ein solches Paar gefunden (Zeile 3 und 4 in Bild 7.19 b), so wird folgendes durchgeführt:

then

$$T := (N, E - \{e_1, e_2\} + \{(e_1 \vee e_2)\}, f', A, B), \text{ wobei } ...(7.8)$$

d. h., die Zeilen mit den Ausdrücken e_1 und e_2 (in unserem Beispiel die Zeilen 3 und 4) werden entfernt. Stattdessen wird eine neue Zeile (3') mit dem Ausdruck $(e_1 \vee e_2)$ eingesetzt (in unserem Beispiel (x3 or x4)), wobei eine neue T-Funktion f' entsteht.

$$f'(e) = f(e) \text{ falls } e \notin \{e_1, e_2\} \text{ und } ....(7.9) \qquad (7.8)$$

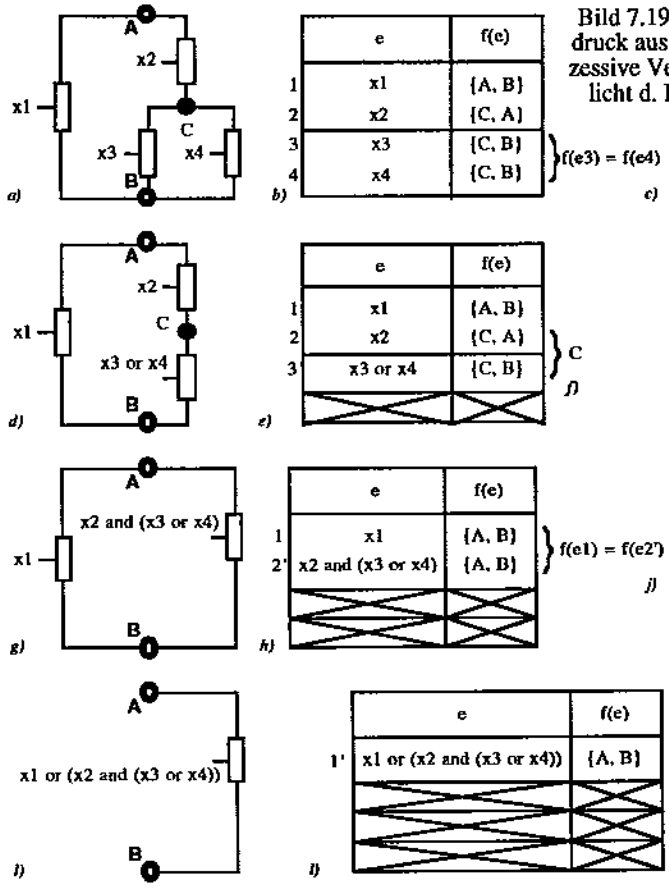


Bild 7.19: Gewinnung eines T-Ausdruck aus einem T-Netzwerk d. sukzessive Verschmelzung, veranschaulicht d. Beispiel n. Abschnitt7.5.2 .

heißt, daß diese in allen denjenigen Zeilen unverändert bleibt, die nicht zu den oben bearbeiteten Ausdrücken e_1 und e_2 gehören. Und der Funktionswert $f(e_1)$ bleibt unverändert und wird somit zum Funktionswert für $f((e_1 \vee e_2))$, in unserem Beispiel für $f((x3 \vee x4))$:

$$f'((e_1 \vee e_2)) = f(e_1); \tag{7.9}$$

Finden wir hingegen keine zwei Ausdrücke e_1 und e_2 mit gleichen Knotenmengen f(e1) und f(e2), so wird ein Knoten X gesucht, der weder Anfangs- noch Endknoten ist (denn letztere dürfen ja nicht eliminiert werden) mit $X \in N - \{A, B\}$, der gleichzeitig zu zwei Ausdrücken e_1 und e_2 gehört mit $e_1, e_2 \in E$.

$$\text{if } \exists\, X \in N - \{A, B\},\ e_1, e_2 \in E:$$

Es wird also ein Knoten X gesucht, an welchem sich zwei in Serie geschaltete Transmissionspfade e_1 und e_2 treffen. Dieser Knoten muß der Vereinigungsmenge $f(e_1)$ und $f(e_2)$ angehören:

$$X \in f(e_1) \cdot f(e_2) \wedge$$

und diese Vereinigungsmenge muß genau drei Knoten umfassen, um sicher zu gehen, daß es sich wirklich um eine Serienschaltung handelt:

$$|f(e_1) \cup f(e_2)| = 3 \wedge$$

und der Knoten X darf nur zu den Mengen der Ausdrücke e_1 und e_2 gehören (um sicher zu gehen, daß an X keine dritten Pfade angeschlossen sind):

$$X \notin \bigcup_{e \in E - \{e_1, e_2\}} f(e) \tag{7.10}$$

Falls ein solcher Knoten X gefunden worden ist (zum Beispiel Knoten C in Bild 7.19 Bild 2.3.2.1c und d), dann wird folgendes durchgeführt: Dieser Knoten wird aus der Gesamtmenge N der Knoten entfernt

then

$$T := (N - \{X\}, E - \{e_1, e_2\} + \{(e_1 \wedge e_2)\}, f', A, B),$$

d. h., die Ausdrücke gefundenen e_1 und e_2 werden entfernt (die Zeilen 2 und 3' in Bild 7.19 f). An deren Stelle tritt der Ausdruck (e_1 and e_2) (der Ausdruck x and (x3 or x4) in Zeile 2` von Bild 7.19 h), wobei aus f eine neue Transmissionsfunktion f' entsteht, derart, daß alle Zeilen gleich bleiben, die nicht zu e_1 und e_2 gehören):

$$f'(e) = f(e) \text{ falls } e \notin \{e_1, e_2\}$$

und die Knotenmenge zum neu gefundenen Ausdruck ($e_1 \wedge e_2$) wird aus durch Vereinigung der beiden Mengen $f(e_1)$ und $f(e_2)$ aus f gebildet unter Fortfall des Knotens X:

$$f'((e_1 \vee e_2)) = f(e_1) \cup f(e_2) - \{X\}$$

All das oben veranschaulichte wird so oft wiederholt, bis keine Änderung der Transmissionsfunktion mehr erreicht werden kann:

until $T = T_{old}$;

Dies ist ein Zeichen dafür, daß alle diejenigen Knoten außer A und B aufgebraucht sind. Die Transmissionstabelle hat nur noch eine Zeile eT_0: der dort stehende Ausdruck eT_0 ist der gesuchte Ausdruck.

if |E| = 1 then write (E)

Deshalb wird er ausgegeben als Resultat. Wenn die Menge E der Ausdrücke jedoch länger als 1 sein sollte, dann ist das Netz nicht serien/parallel zerlegbar. Dies wird diagnostiziert durch:

else write (' e_{T0} nicht serien parallel zerlegbar')

end.

Die Zerlegungsprozedur ist nun beendet.

7.5.3 Ermittlung eines Transmissionsnetzes aus einem T-Ausdruck

Dieser Abschnitt führt einen Algorithmus zur Ermittlung eines Transmissionsnetzes aus einem Transmissionsausdruck ein. Der Algorithmus ist rekursiv.

Eingabe: Transmissionsausdruck e, Anfangsknoten A und Endknoten B

Ausgabe: Transmissionsnetz T_e

Verfahren:

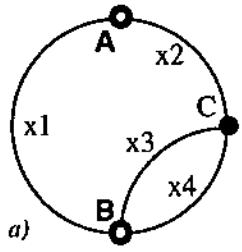
```
procedure generate (in e, A, B; out T);
begin
  if e = (e1 ∧ e2) then
    begin
      sei C ein "neuer Knoten";
      generate (e1, A, C, T1);
      generate (e2, C, B, T2);
      seien T1 = (N1, E1, f1, A, C) und T2 = (N2, E2, f2, C, B);
      T := ( N1∪N2, E1∪E2, f1∪f2, A, B)
    end
  else if e = (e1 ∨ e2) then begin
      generate (e1, A, B, T1);
      generate (e2, A, B, T2);
      seien T1 = (N1, E1, f1, A, B) und T2 = (N2, E2, f2, A, B);
      T := ( N1∪N2, E1∪E2, f1∪f2, A, B)
    end
  else T := ({A, B},{e}, f, A, B), wobei f(e) = { A,B }
end.
```

Erläuterung. Es werden drei Fälle unterschieden:

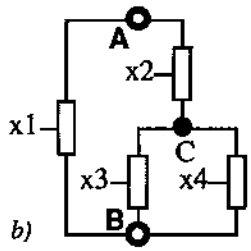
1) Der Ausdruck beschreibt eine Parallelschaltung von zwei Transmissionsnetzen, die den Ausdrücken e_1 und e_2 zugeordnet sind. Zunächst werden diese Netze durch rekursive Aufrufe bestimmt und danach zu einem einzigen Netz zusammengebaut.

1) Der Ausdruck beschreibt eine Serienschaltungen von zwei Transmissionsnetzen, die den Ausdrücken e_1 und e_2 zugeordnet sind. Zunächst werden diese Netze durch rekursive Aufrufe bestimmt und danach zu einem einzigen Netz zusammengebaut. Dabei wird ein neuer (bisher nirgends vorkommender Knoten) benötigt.

1) Es handelt sich um einen atomaren Ausdruck (Eingangsleitung). In diesem Fall liefert der Algorithmus ein Transmissionsnetz, das nur aus einer einzigen, mit der Eingangsleitung markierten Kante besteht.



a)



b)

c)

e	f(e)
x1	{A, B}
x2	{C, A}
x3	{C, B}
x4	{C, B}

Bild 7.20: T-Netz-Notationen a) Transmissions-Graph, b) Schalterrealisierung, c) Tabelle.

Beispiel: Für den Transmissionsausdruck (x1 **or** (x2 **and** (x3 **or** x4))) liefert der Algorithmus als Ausgabe das Transmissionsnetz $T = (\{A, B, C\}, \{x1, x2, x3, x4\}, f, A, B)$, mit $f(x1) = \{A, B\}$, $f(x2) = \{C, A\}$, $f(x3) = \{C, B\}$ und $f(x4) = \{C, B\}$ das graphisch wie in Bild 7.20 dargestellt werden kann.

7.6 Nicht-Serien/Parallel zerlegbare Schalternetze

Wir haben uns bis jetzt nur mit solchen logischen Zweipolen beschäftigt, die durch eine Parallel/Serien-Zerlegung (Bild 7.15) vollständig analysiert werden konnten. Dies ist jedoch nur ein Spezialfall unter den logischen Zweipolen, im folgenden Abschnitt behandeln wir ein allgemeineres formales Verfahren, das geeignet ist für alle Arten von Netzen, also auch für nicht serien/parallel zerlegbare Schaltungen [5] [MUR79]. Im Fall der serien/parallel zerlegbaren Netze konnten wir irgendein Parallel-Netz T_p in zwei parallel geschaltete Subnetze T_q und T_r (Bild 7.21 a) derart zerlegen, daß sich die logische Gesamtfunktion T_p ergibt (Bild 7.21 b) mit $T_p = T_q$ **or** T_r (Parallelzerlegung). Dann hatten wir noch den Fall der Serienzerlegung. Wir können ein Schaltnetz T_s (Bild 7.21 c) zerlegen in die Serienschaltung von zwei Subnetzen T_t und T_u, (Bild 7.21 d) wobei sich dann die Gesamt-Transmissionsfunktion T_s, d.h. die Durchlässigkeit zwischen den Anschlüssen A und B, ergibt mit: $T_s = T_t$ **and** T_u.

Nicht-Serien/Parallel zerlegbare Schalternetze sind solche, bei denen das in Abschnitt 7.5 vorgestellte Verfahren versagt. Deswegen benötigen wir ein anderes Verfahren. Es gibt zwei Methoden: erstens die Ermittlung sogenannter Verbindungsmengen, auf englisch *tie sets* genannt, und zweitens eine dazu duale Methode: die Ermittlung der sogenannten Schnittmenge, oder auf englisch *cut set* genannt. Anhand eines Beispieles wird nun im einzelnen erklärt, was darunter zu verstehen ist.

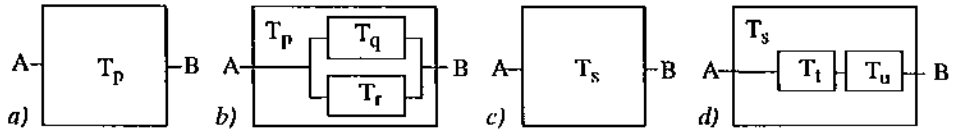


Bild 7.21: Logische Zweipole mit Parallelschaltung(a,b) und Serienschaltung (c, d).

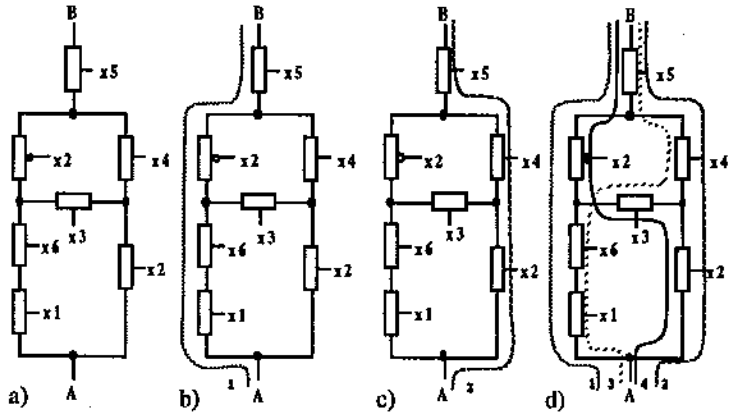


Bild 7.22: Logischer Zweipol zur Ermittlung der Bindemenge.

Als Beispiel zur Veranschaulichung des Verfahrens der Verbindungsmengen sei der logische Zweipol in Bild 7.22 a verwendet. Zunächst wird die folgende Definition eingeführt:

Definition "Linearer Pfad" in einem logischen Zweipol:

> Der Pfad verbindet die beiden äußeren Anschlüsse A und B des Zweipols miteinander (wie in Bild 2.4.2 b). Der Pfad darf keine Schleifen enthalten.

Definition "Verbindungsmenge":

> Die Menge aller Schaltervariablen, welche die Schalter längs eines linearen Pfades repräsentieren, werden dann *Verbindungsmenge* (oder engl.: *tie set*) dieses logischen Zweipols genannt.

Im Beispiel nach Bild 7.22 b ist die Verbindungsmenge V1 = {x1, x6, **not** (x2), x5}. Eine zweiter Verbindungs-Pfad innerhalb des Zweipol ist in Bild 7.22 c zu sehen. Die zu diesem Pfad gehörige Verbindungsmenge ist V2 = {x2, x4, x5}. Zwischen den beiden äußeren Anschlüsse a und b spannen sich noch zwei weitere Pfade Nr. 3 und Nr. 4 auf mit V3 = {x1, x6, x3, x4, x5} und V4 = {x2, x3, **not** (x2), x5}. Bild 7.14 d zeigt sämtliche 4 Pfade gleichzeitig, die entsprechende Liste sämtlicher Verbindungsmengen ist:

V1 = {x1, x6, **not** (x2), x5}

V2 = {x2, x4, x5}

V3 = {x1, x6, x3, x4, x5}

V4 = {x2, x3, **not** (x2), x5}

Das sind also die 4 Verbindungsmengen, die dieses Schaltnetz enthält. Wir gewinnen jetzt daraus die Verhaltensbeschreibung, die Beschreibung in Form einer Boole'schen Funktion, indem

man aus jeder Verbindungsmenge eine Konjunktion bildet. Die Boole'sche Gesamtfunktion ergibt sich dann aus der Disjunktion aller dieser Konjunktionen.

T = x1 x6 not(x2) x5
or x2 x4 x5
or x1 x6 x3 x4 x5
or x2 x3 not(x2) x5

Natürlich kann man jetzt noch eine Optimierung durchführen. Das gehört aber zur digitalen Logik, die nicht Gegenstand dieses Textes ist (s. z. B. [5]). Jetzt kommen wir zu der zweiten Möglichkeit, die zur eben gezeigten Methode dual ist, nämlich die Ermittlung der Schnittmengen. Zunächst benötigen wir noch die folgende Definition:

Definition "Schnittmenge":

Die Schnittmenge (*cut set*) ist eine Menge, die aus Kontakten eines logischen Zweipols besteht (Beispiel: S = {x1, x2}). Jede Schnittmenge muß folgende zwei Bedingungen erfüllen:

1. Sind alle Kontakte der Schnittmenge undurchlässig, ist der Zweipol in zwei voneinander isolierte Subnetze zerlegt, d.h. es existiert keine durchlässige Verbindung zwischen den beiden Anschlüssen des Zweipols.
2. Sobald mindestens einer der Kontakte durchlässig ist, sind die beiden Subnetze miteinander verbunden, d.h. es existiert eine durchlässige Verbindung zwischen den beiden Anschlüssen des Zweipols.

Als Beispiel diene der logische Zweipol in Bild 7.23 a. Bild 7.23 b zeigt eine Schnittmenge hierzu mit S = {x1, x2}. Eine zweite Schnittmenge wird veranschaulicht durch Bild 7.23 c. Zu dieser Schnittmenge gehören x6 und x2, und somit ergibt sich S2 = {x2, x6}. Eine dritte Schnittmenge wird gezeigt durch Bild 7.23 d. Sie geht quer durch x1, x3, x4, womit S3 = {x1, x3, x4} ist. Die Bild 7.23 e bis h veranschaulichen sämtliche weiteren Schnittmengen S4 bis S7 zum Beispiel in Bild 7.23 a. Bild 7.23 j ist eine zusammenfassende Darstellung, in welcher alle 7 Schnittmengen gezeigt sind, die zu diesem logischen Zweipol - CS-Netz - gehören, entsprechend folgender Liste:

S1 = {x1, x2}
S2 = {x6, x2}
S3 = {x1, x3, x4}
S4 = {x6, x3, x4}
S5 = { not (x2), x3, x2}
S6 = { not (x2), x4}
S7 = {x5}

Wir gewinnen jetzt aus den Schnittmengen dieses logischen Zweipols die Verhaltensbeschreibung in Form einer Boole'schen Funktion, indem wir innerhalb der jeweiligen Schnittmengen

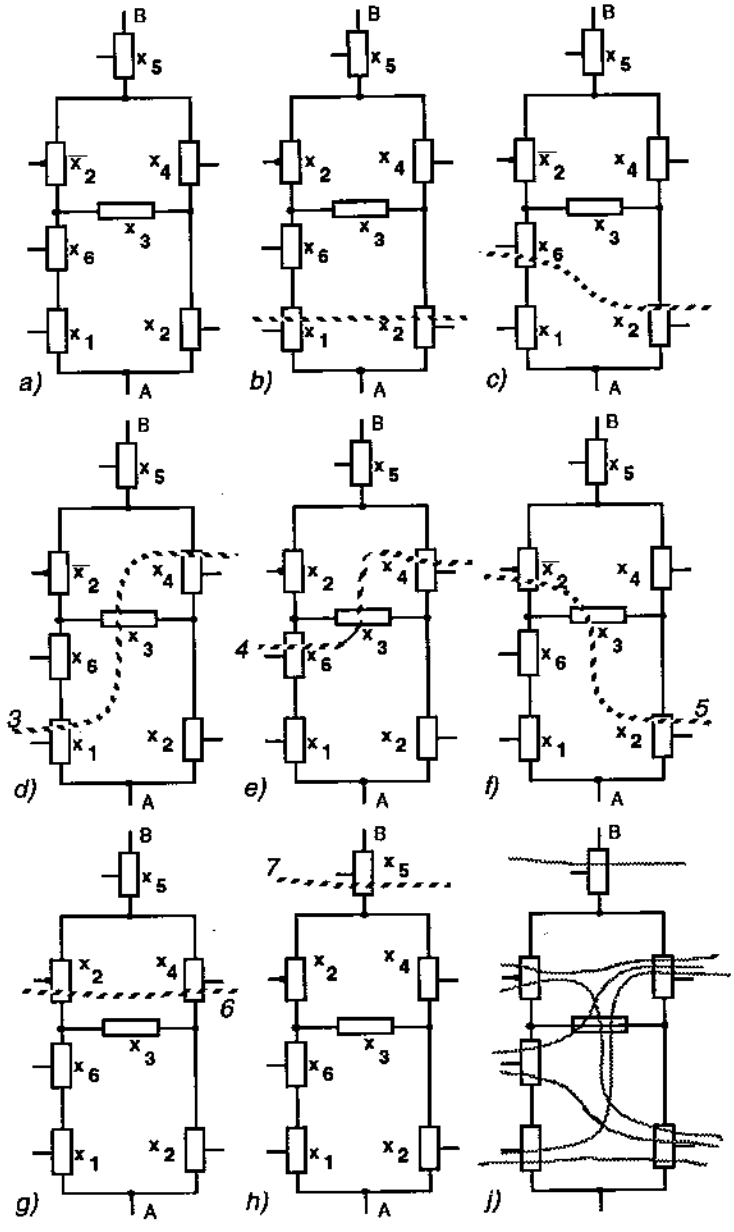


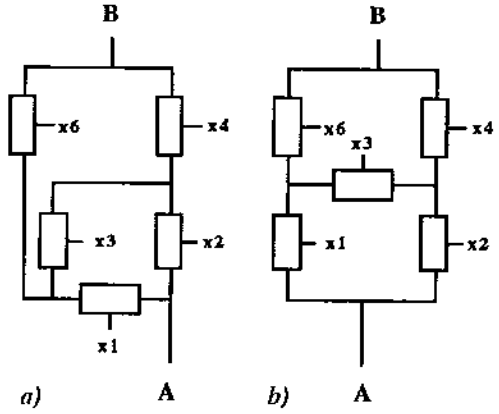
Bild 7.23: Logischer Zweipol: Gewinnung der Verhaltensbeschreibung über die Bildung der Schnittmenge

die Variablen mit or verknüpfen. Die jeweils daraus gewonnen Disjunktionen verknüpfen wir dann konjunktiv miteinander. Das Ergebnis ist das folgende:

f = (x1 or x2) and (x6 or x2) and (x1 or x3 or x4)
(x6 or x3 or x4) and (not (x2) or x3 or x2)
(not (x2) or x4) and (x5).

Wir haben also zusätzlich zur Serien/Parallel-Zerlegung zwei weitere Methoden zur Analyse logischer Zweipole kennengelernt. Die letzteren Methoden können wir immer anwenden, auch wenn die Serien/Parallel-Zerlegung versagt. Zur Ermittlung der Serien/Parallel-Zerlegbarkeit wird hier keine formale Methode angegeben. (Der Algorithmus von Abschnitt 7.5.2 ist eigentlich eine formale Methode: er liefert entweder denT-Ausdruck oder die Meldung "nicht seriel/parallel zerlegbar".)

Oft kann man die Serien/Parallel-Zerlegbarkeit an der graphischen Darstellung intuitiv sehr rasch entscheiden. Eine weniger geschickte Anordnung der graphischen Darstellung kann jedoch hierbei zu Täuschungen führen. Als Beispiel zur Veranschaulichung soll Bild 7.24 a dienen: Ist dieser logische Zweipol Serien/Parallel-zerlegbar? In Bild 7.24 b ist dies leichter zu erkennen als im ursprünglichen Bild 7.24 a. In Bild 7.24 b ist die Brücken-Funktion des Schalters x3 deutlicher zu erkennen. Solche überbrückenden Schalter, die eine Querverbindung zwischen zwei Serien-Pfaden herstellen, machen eine Serien/Parallel-Zerlegung unmöglich. Also dieses Netz ist nicht serien/parallel-zerlegbar.



a) b)

Bild 7.24: Darstellungen eines nicht Serien-Parallel-zerlegbaren Schalternetz-Beispiels: a) mit verborgener Brücke, b) mit deutlich sichtbarer Brücke.

Nun sei ein zweites Beispiel auf Serien-Parallel-Zerlegbarkeit hin untersucht (Bild 7.25). Wie oben gezeigt, sind die Netze dann nicht serien/parallel-zerlegbar, wenn sie einen Querpfad haben, der zwei parallel verlaufende Pfade miteinander verbindet. In Bild 7.25 a ist ein solcher Querpfad zu sehen. Heißt dies, daß

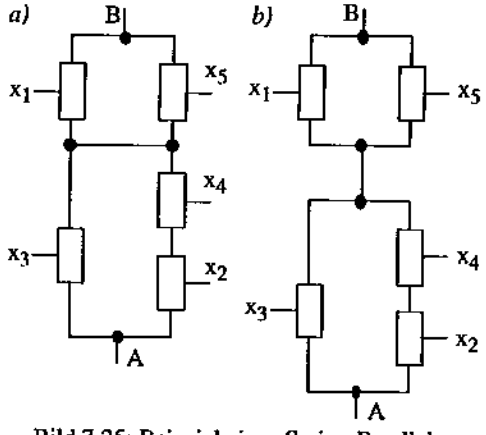


Bild 7.25: Beispiel eines Serien-Parallel zerlegbaren Zweipols: a) mit vorgetäuschter Brücke, b) in transparenter Darstellungsweise.

Serien/Parallel-Zerlegbarkeit nicht gegeben ist? Wenn die graphische Darstellung topologisch (jedoch nicht funktional) geändert wird gemäß Bild 7.25 b, erhält man mehr Transparenz der Situation. Die Querverbindung kann in eine doppelte Gabel umgewandelt werden, da die Querverbindung nur aus Verdrahtung besteht, jedoch keinerlei "Bauelemente" enthält. Das Netz ist also Serien/Parallel-zerlegbar.

7.6.1 Superposition logischer Zweipole

Im Folgenden betrachten wir eine etwas kompliziertere Schaltung. Das Netz in Bild 7.26 ist kein logischer Zweipol. Es können jedoch zwei verschiedene logische Zweipole daraus isoliert werden. Ein logischer Zweipol Z1 geht von A nach B, ein anderer (Z2) von A nach C. Wenn man die Schalter x1 und not(x2) wegläßt, dann hat der Zweipol Z1 trotzdem noch das gleiche Verhalten wie zuvor. Hier ist also eine Vereinfachung möglich, beim logischen Zweipol Z2 jedoch nicht. Sämtliche Kontakte nach Bild 7.26 sind nötig, um das Verhalten des Zweipols Z2 vollständig zu realisieren. Man sieht also: das Handwerkszeug, das wir bisher betrachtet haben, kann auch für komplexere Gebilde verwendet werden, die sich aus der Überlagerung von mehreren logischen Zweipolen bilden lassen.

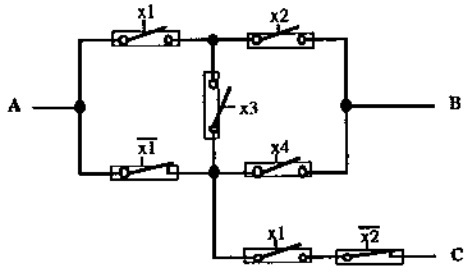


Bild 7.26: Überlagerung zweier Zweipole

Schaltnetz-Probleme mit Kontext. Wir können mit den Verfahren, die bisher eingeführt wurden, noch eine Reihe andersartiger Aufgaben lösen, wie beispielweise diejenigen aus Bild 7.27. Hier wird nicht nur die reine Durchlässigkeit eines logischen Zweipoles analysiert. Vielmehr wird der Fluß durch ein in das Netz eingebettetes andersartiges Objekt (z.B. ein Verbraucher) untersucht. Bild 7.27 a zeigt nun einen Zweipol, der zusätzlich zu den Schaltern noch eine Lampe enthält. Bei der Lösung des Problems, wann diese Lampe "brennt" oder nicht, muß man sich vorstellen, daß zwischen A und B eine Spannung eingespeist wird. Es müssen nun diejenigen Pfade gefunden werden, die über die Lampe von A nach B führen. Dann werden die Konjunktionen gemäß den Verbindungsmengen daraus gebildet und diese wiederum disjunktiv miteinander verknüpft. Zusätzlich müssen noch die Fälle berücksichtigt werden, in welchen die Lampe kurzgeschlossen wird, und deshalb auch bei Pfaddurchlässigkeit nicht brennen kann.

Man kann also das Verfahren, das wir bisher kennengelernt haben, nicht unmittelbar anwenden. Eine Variante muß entwickelt werden, bei der noch Zusatzbedingungen beachtet werden müssen. Ähnlich dem Beispiel in Bild 7.26 muß das Netz in die Superposition zweier Teilnetze T1 und T2 umgewandelt werden, die nicht notwendigerweise disjunkt sein müssen. T1 entspricht T(A,B) (d. h. es ist zwischen den Anschlüssen A und B aufgespannt) und T2 entspricht T(C,B). Die Lampe brennt dann, wenn

$$f = T1 \text{ and not } (T2) = 1$$

ist (d. h., wenn T1 die Lampe versorgt und T2 diese nicht kurzschließt). Wie sich leicht zeigen läßt, gilt für die Aktivierung der Lampe in Bild 7.27 b die Boole'sche Funktion f mit:

$$f = T1 \text{ and not } (T2) \text{ and } T3 = 1,$$

wobei T1 ::= T(A,C), T2 ::= T(B,C) und T3 ::= T(C,D). In diesem Fall haben wir eine Superposition aus drei Subnetzen, wobei T1 und T3 zur Versorgung der Lampe nötig sind und T2 diese nicht kurzschließen darf.

7.7 Literatur

[1] S. H. Caldwell: Switching Circuits and Logic Design; Wiley, 1958

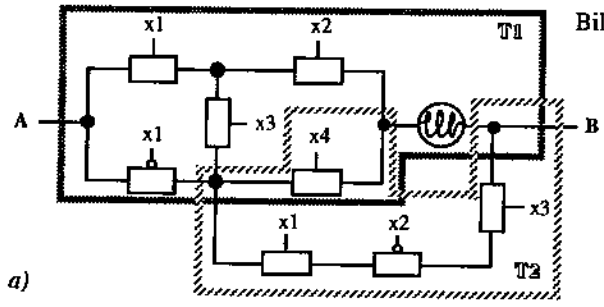
[2] R. W. Hartenstein: KARL-3 Reference Manual; Univ. Kaiserslautern, 1986

[3] R. Hauck: KARL-3 User Guide; Univ. Kaiserslautern, 1986

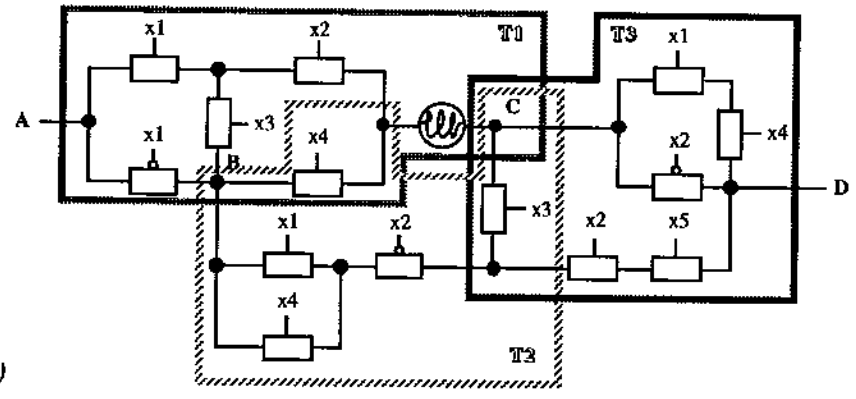
[4] A, Mukherjee: nMOS & CMOS VLSI System Design; Prentice-Hall, 1986

[5] S. Muroga: Logic Design and Switching Theory; Wiley & Sons, 1979

[6] C. E. Shannon: The Synthesis of Two-Terminal Switching Circuits; Bell Syst. Techn. Journal 28,1, 1949



a)



b)

Bild 7.27: Zweipole mit Verbraucher