10.1.4 Aktive Filter

Während bei höheren Frequenzen (oberhalb 100 kHz) Filter aller gewünschten Charakteristika (sog. kritische, Bessel, Butterworth, Tschebyscheff, s. Tietze u. Schenk (1991) S. 391ff.) leicht mit Widerständen, Kondensatoren und Spulen aufgebaut werden können, ist im Niederfrequenzbereich die Realisierung der Spulen wegen der erforderlichen großen Induktivitätswerte schwierig.

Unter Verwendung von Operationsverstärkern mit einer äußeren Beschaltung lediglich aus Widerständen und Kondensatoren lassen sich dagegen auch für sehr niedrige Frequenzen (z.B. 50 Hz) alle Typen von Filtern problemlos aufbauen. Fig. 10.16 zeigt ein typisches Tiefpaßfilter zweiter Ordnung mit dem Frequenzgang

$$A\left(\mathrm{j}\frac{\omega}{\omega_\mathrm{g}}\right) = -\frac{R_2/R_1}{1+\mathrm{j}\omega C_2\left(R_2+R_3+\dfrac{R_2\cdot R_3}{R_1}\right)-\omega^2 C_1 C_2 R_2 R_3} \tag{10.20}$$

wobei $\omega_\mathrm{g}/2\pi$ die gewünschte Grenzfrequenz des Filters ist.

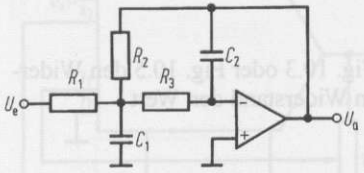


Fig. 10.16
Aktives Tiefpaßfilter
U_e Eingangsspannung
U_a Ausgangsspannung

Aus einem Koeffizientenvergleich mit der allgemeinen Gleichung für den Frequenzgang eines Tiefpasses zweiter Ordnung

$$A\left(\mathrm{j}\frac{\omega}{\omega_\mathrm{g}}\right) = \frac{A_0}{1+\mathrm{j}\dfrac{\omega}{\omega_\mathrm{g}}a-\left(\dfrac{\omega}{\omega_\mathrm{g}}\right)^2 b} \tag{10.21}$$

lassen sich die Werte der einzelnen Bauelemente berechnen, wozu zweckmäßigerweise ein programmierbarer Taschenrechner benutzt wird. Die Faktoren a und b hängen vom gewählten Filtertyp ab. Sie sind in der Literatur tabellarisch angegeben (s. Tietze u. Schenk (1991) S. 405ff.).

Literatur zu 10.1

Tietze, U.; Schenk, Ch. (1991) Halbleiter-Schaltungstechnik 9. Aufl. Berlin-Heidelberg-New York: Springer

10.2 Grundlagen der Digitaltechnik (P. Seyfried †)

Die zunächst der Datenverarbeitung, Speicherung und Übertragung vorbehaltene Digitaltechnik dringt mit der Entwicklung immer komplexerer Systeme mehr und mehr in Gebiete vor, die bisher der Analogtechnik vorbehalten waren. Z.B. Regelungstechnik, Schallaufzeichnung, Bildaufzeichnung, Sprachsynthese. Hier werden nur einige, der Realisierung von

Grundfunktionen dienende Schaltungen näher beschrieben. Die weiteren Ausführungen betreffen Probleme, die bei der Anwendung digitaler Geräte im Laboratorium häufig auftreten.

Zusammenfassende Literatur: Haack (1984); Lancaster (1980); Leonhardt (1976); Steinbuch (1962 u. 1974); Texas Instrum. (1973 u.1977); Tietze u. Schenk (1980); Valvo (1980); Zuiderveen (1981).

10.2.1 Dualzahlen

Eine digitale Schaltung wird durch die Abhängigkeit ihrer Ausgangsvariablen Y_k von ihren Eingangsvariablen X_i funktionell beschrieben. Die Variablen sind im allgemeinen binäre Signale mit zwei bedeutungsvollen Zuständen (vorzugsweise Spannungen), denen die Binärziffern „Null" und „Eins" (Symbole 0 bzw. 1) zugeordnet werden. Bei positiver Logik (activ high) wird der positivere der zwei Signalzustände (high level, H), bei negativer Logik (activ low) der negativere (low level, L) der „Eins" zugeordnet. Ein binäres Signal stellt die kleinste (zweiwertige) Informationseinheit (das Bit) dar. Mehrere Bits lassen sich zu Zahlen unterschiedlicher Kodierung zusammenfassen. Bei der rein dualen Zahlendarstellung entspricht jedes Bit einer Stelle der Dualzahl, die sich in der Form

$$Z_{\text{dual}} = \sum_{i=1}^{n} \alpha_i 2^{i-1} \tag{10.22}$$

mit $\alpha_i = 0$ oder 1 darstellen läßt.

Werden drei binäre Stellen zusammengefaßt und den acht möglichen Kombinationen die Werte β_i zwischen 0 und 7 zugeordnet, so kommt man zur oktalen Zahlendarstellung

$$Z_{\text{okt}} = \sum_{i=1}^{n} \beta_i (2^3)^{i-1} \tag{10.23}$$

Bei der sedezimalen (häufig auch hexadezimal genannt) Zahlendarstellung bilden vier Bits die 16 Ziffern γ_i, die üblicherweise mit den Ziffern 0 bis 9 ergänzt durch die Buchstabenfolge A bis F geschrieben werden:

$$Z_{\text{hexa}} = \sum_{i=1}^{n} \gamma_i (2^4)^{i-1} \tag{10.24}$$

Um Daten digital zu erfassen und zu verarbeiten, wird fast ausschließlich der reine Dualkode verwendet, wegen dessen schlechter Lesbarkeit aber zur Ausgabe der Oktalkode und Sedezimalkode vorgezogen, wenn nicht der Dezimalkode verwendet wird, bei dem von den 16 möglichen Kombinationen aus vier Bits nur 10 ausgenutzt werden.

10.2.2 Grundfunktionen

Von den grundlegenden Funktionen zwischen logischen Variablen x_i (Steinbuch (1962), S. 105) sind hier die folgenden von besonderer Bedeutung:

Negation	$y = \overline{x}$	(10.25)
Konjunktion	$y = x_1 \cdot x_2 = x_1 x_2$	(10.26)
Disjunktion	$y = x_1 + x_2$	(10.27)

Für die Verknüpfungen gelten folgende Regeln der Booleschen Algebra (Steinbuch (1962), S. 112):

Kommutatives Gesetz:

$$x_1 x_2 = x_2 x_1 \qquad x_1 + x_2 = x_2 + x_1 \tag{10.28),(10.29}$$

Assoziatives Gesetz:

$$x_1(x_2 x_3) = (x_1 x_2) x_3 \tag{10.30a}$$

$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3 \tag{10.30b}$$

Distributives Gesetz:

$$x_1(x_2 + x_3) = x_1 x_2 + x_1 x_3 \tag{10.31a}$$

$$x_1 + x_2 x_3 = (x_1 + x_2)(x_1 + x_3) \tag{10.31b}$$

Absorptionsgesetz:

$$x_1(x_1 + x_2) = x_1 \qquad x_1 + x_1 x_2 = x_1 \tag{10.32a),(10.32b}$$

Tautologie:

$$xx = x \qquad x + x = x \tag{10.33a),(10.33b}$$

Negation:

$$x\overline{x} = 0 \qquad x + \overline{x} = 1 \tag{10.34a),(10.34b}$$

Doppelte Negation :

$$\overline{(\overline{x})} = x \tag{10.35}$$

De Morgans Gesetz:

$$\overline{x_1 x_2} = \overline{x}_1 + \overline{x}_2 \qquad \overline{x_1 + x_2} = \overline{x}_1 \overline{x}_2 \tag{10.36a),(10.36b}$$

Operationen mit 0 und 1:

$$x \cdot 1 = x \qquad x + 0 = x \tag{10.37a),(10.37b}$$

$$x \cdot 0 = 0 \qquad x + 1 = 1 \tag{10.38a),(10.38b}$$

$$\overline{0} = 1 \qquad \overline{1} = 0 \tag{10.39a),(10.39b}$$

Weiterhin sind folgende Funktionen zu beachten, die sich auf die eingangs genannten Grundfunktionen zurückführen lassen:

$$\text{Antivalenz} \quad y = x_1 \oplus x_2 \quad \text{d.h.} \quad y = \begin{cases} 0, & \text{wenn } x_1 \equiv x_2 \\ 1, & \text{wenn } x_1 \not\equiv x_2 \end{cases} \tag{10.40}$$

$$\text{äquivalenz} \quad y = \overline{x_1 \oplus x_2} \quad \text{d.h.} \quad y = \begin{cases} 1, & \text{wenn } x_1 \equiv x_2 \\ 0, & \text{wenn } x_1 \not\equiv x_2 \end{cases} \tag{10.41}$$

Heim (1973).

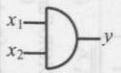
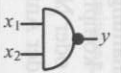
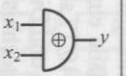
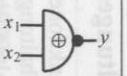
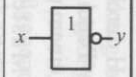
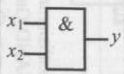
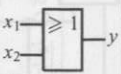
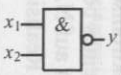
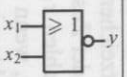
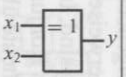
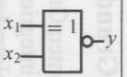
Funktion	Eingangsvariable	Negation	Eingangsvariable		Konjuktion	Disjunktion	NAND	NOR	Antivalenz	Äquivalenz
Sprachliche Bezeichnung		NICHT			UND	ODER			EXOR	EXNOR
Schreibweise der Schaltalgebra	x	$y = \overline{x}$	x_1	x_2	$y = x_1 \cdot x_2$	$y = x_1 + x_2$	$y = \overline{x_1 \cdot x_2}$	$y = \overline{x_1 + x_2}$	$y = x_1 \oplus x_2$	$y = \overline{x_1 \oplus x_2}$
Wahrheitstabelle	0	1	0	0	0	0	1	1	0	1
			0	1	0	1	1	0	1	0
	1	0	1	0	0	1	1	0	1	0
			1	1	1	1	0	0	0	1
altes Symbol										
Symbol nach IEC 617-12										

Fig. 10.17 Grundfunktionen für zwei Eingangs-(x_1, x_2) und eine Ausgangsvariable (y); Wahrheitstabelle und Schaltungssymbole nach IEC 617-12; dort auch Symbole für Mehrfacheingänge

10.2.3 Grundschaltungen

Die genannten Grundfunktionen werden durch Grundschaltungen realisiert, die heute in der Regel als integrierte Schaltung aufgebaut sind und als Gatter oder Tore bezeichnet werden. In allen folgenden Schaltungsbeispielen sind die Versorgungsspannungen (U_{CC}, U_{DD}, Masse) üblicherweise nur dann eingezeichnet, wenn sie zur Funktion der äußeren Beschaltung notwendig sind. In Fig. 10.17 sind die vorgenannten Grundfunktionen, ergänzt durch die Negation von Konjunktion und Disjunktion (NAND, NOR) zusammengestellt. Zu jeder Funktion ist die heute übliche technische Kurzbezeichnung, die Schreibweise in der Schaltalgebra, das Schaltzeichen nach IEC 617-12 angegeben. In der jeweiligen sog. Wahrheitstabelle ist eindeutig der Zustand der Ausgangsvariablen y für alle möglichen Kombinationen der Eingangsvariablen x_1, x_2 abzulesen. So nimmt y bei der Konjunktion dann und nur dann den Wert 1 an, wenn sowohl x_1 als auch x_2 den Wert 1 aufweisen. Von besonderer Bedeutung für die elektronische Schaltungstechnik sind die NAND- und NOR-Verknüpfungen, da jede Funktion eines beliebig komplizierten Schaltnetzes unter Anwendung der angegebenen Regeln allein auf NAND- oder NOR-Verknüpfungen für zwei Variable zurückgeführt werden kann. Die Realisierung der Antivalenz (EXOR) durch NAND-Gatter zeigt Fig. 10.18.

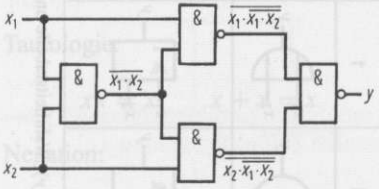


Fig. 10.18 Realisierung der EXOR-Funktion durch NAND-Gatter
$y = x_1 \oplus x_2 = x_1 x_2 \cdot (x_1 + x_2)$

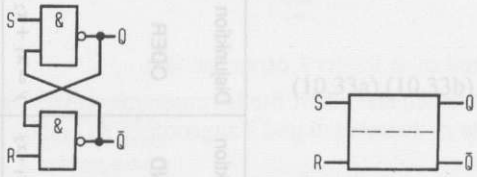


Fig. 10.19 RS-Flip-Flop aus NAND-Gattern und Schaltsymbol nach DIN 40 700 Tl. 14 (1976)
S Setzeingang, R Rücksetzeingang,
Q und $\overline{Q}$ Ausgänge

Die wichtigste Funktionsgruppe für Speicher ist die bistabile Kippstufe, auch Flip-Flop (Kurzbez. FF) genannt, die einmal in einen Zustand geschaltet, diesen so lange beibehält bis sie durch ein folgendes Signal in den anderen umschaltet. Fig. 10.19 zeigt die einfachste Form eines Rücksetz-Setz-Flip-Flop (kurz RS-FF), der ausgehend vom Ruhezustand S = R = H durch ein Low-Signal am S-Eingang in den Zustand Q = H, $\overline{Q}$ = L und durch einen Low-Pegel an R nach Q = L, $\overline{Q}$ = H schaltet. Gleichzeitige Low-Signale an R und S führen zu einem undefinierten Zustand, aus dem der FF in eine Vorzugslage kippt.

Beim RS-Master-Slave-Flip-Flop (Fig. 10.20 u. Tab. 10.1a) sperrt die positive Flanke des an C angelegten Taktimpulses die Kopplung zwischen Master- und Slave-Teil. Dann öffnet sie die am Eingang des Master-FF liegenden Tore und setzt damit den Master-FF entsprechend der an R und S liegenden Information. Die negative Flanke schließt die Eingangstore wieder und öffnet die Koppeltore zum Slave-FF, der dadurch die Information des Master-FF übernimmt.

High-Pegel an R und S führt wie im vorhergehenden Beispiel zu einem undefinierten Schaltzustand. Der Nachteil wird durch die gestrichelt gezeichneten Verknüpfungen mit den Ausgängen beseitigt (Fig. 10.20 u. Tab. 10.1b). Aus dem RS-FF wird ein JK-Master-Slave-Flip-Flop. Jetzt wird nur das Eingangstor geöffnet, das an seinem dritten Eingang vom Ausgang her einen High-Pegel erhält. Für J = K = H schaltet der FF nach jedem Taktimpuls in den entgegengesetzten Schaltzustand, er untersetzt die Taktfrequenz auf die Hälfte.

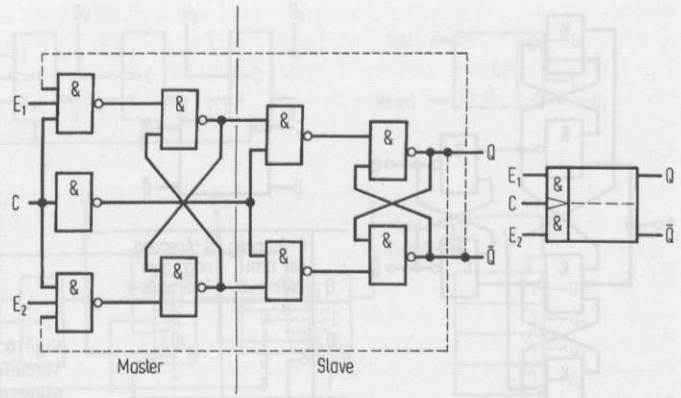


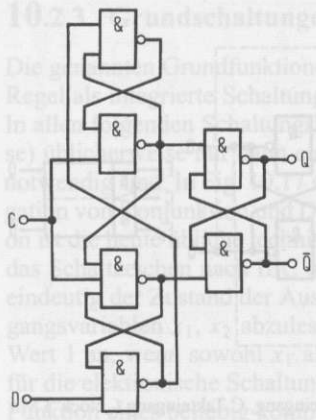
Fig. 10.20 RS-Master-Slave-Flip-Flop E_1 = S Setzeingang, E_2 = R Rücksetzeingang, C Takteingang („clock") Mit Rückführungen (gestrichelt): JK-Master-Slave FF
E_1 = J und E_1 = K Vorbereitungseingänge, C Takteingang („clock")

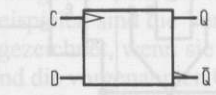
Tab. 10.1 Wahrheitstabellen für die Schaltung nach Fig. 10.20

a) RS-FF				b) JK-FF			
S	R	Ausgang Q vor dem Taktimpuls	Ausgang Q nach dem Taktimpuls	J	K	Ausgang Q vor dem Taktimpuls	Ausgang Q nach dem Taktimpuls
0	0	1	1	0	0	0	0
		0	0			1	1
1	0	1	1	1	0	0	1
		0	1			1	1
0	1	1	0	0	1	0	0
		0	0			1	0
1	1	1 oder 0	unbestimmt	1	1	0	1
						1	0

Bei diesen Master-Slave-Flip-Flop-Stufen darf während der H-Phase des Taktsignals der Pegel an R und S bzw. J und K nicht von H → L schalten. Nichtbeachtung, ein häufiger Fehler bei Schaltungsentwürfen, führt zu Fehlfunktionen. Beim flankengetriggerten JK-Flip-Flop tritt diese Fehlerquelle nicht auf. Hier ist für die Schaltrichtung nur maßgebend, welcher Pegel an J und K zum Zeitpunkt der auslösenden Flanke (positiv flankengetriggert oder negativ flankengetriggert) anliegt.

Das flankengetriggerte D-Flip-Flop, das als Datenübernahme-Speicher („Latch") benutzt wird, besitzt anstelle der zwei Eingänge J und K nur einen Eingang, der meist mit D bezeichnet wird. Mit der auslösenden Flanke des an C angelegten Taktimpulses nimmt der Ausgang Q den an D anliegenden Pegel an (s. Fig. 10.21). Der Erzeugung von Impulsen vorgegebener Dauer dient die monostabile Kippstufe (Univibrator), die sich aus dem FF der Schaltung nach Fig. 10.19 durch Einfügen einer kapazitiven Kopplung ergibt (s. Fig. 10.22a). Durch ein kurzes Low-Signal am Eingang E schaltet die Stufe aus dem stabilen in einen zweiten metastabilen Zustand, in dem sie für eine Zeit $t_0 \approx \tau = RC$ verharrt. Beim Univibrator mit fester Schaltzeit wird die Dauer des Ausgangssignals nicht be-





D	Q-Ausgang vor dem Taktimpuls	Q-Ausgang nach dem Taktimpuls
0	0	0
	1	0
1	0	1
	1	1

Fig. 10.21
Vereinfachtes Schaltbild eines flankengetriggerten D-Flip-Flop mit Wahrheitstabelle

einflußt, wenn während dieser Zeit weitere Signale an E auftreten. Beim Univibrator mit verlängerbarer Schaltzeit (retriggerable univibrator) wird das Ausgangssignal durch jeden Triggerimpuls um den gleichen Betrag verlängert.

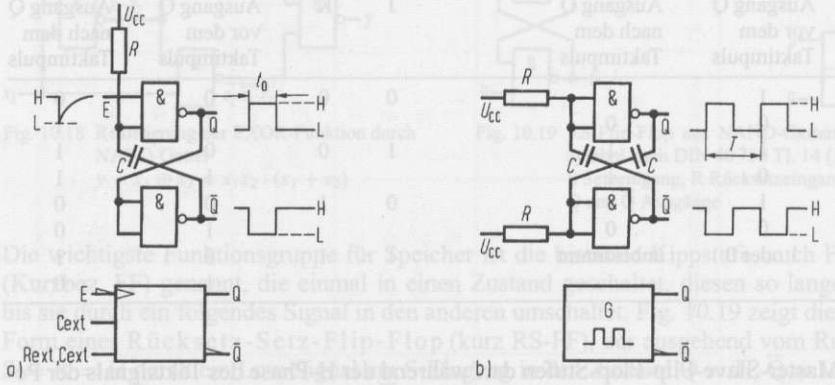


Fig. 10.22 a) Monostabile Kippstufe (Univibrator) b) astabile Kippstufe (Multivibrator)
U_{CC} = Versorgungsspannung

Fügt man in der Schaltung nach Fig. 10.22a eine zweite kapazitive Kopplung ein, so besitzt die Schaltung (Fig. 10.22b) keinen stabilen Schaltzustand mehr. Sie kippt ständig von dem einen metastabilen Zustand in den anderen. Die Frequenz $f_0 \approx 1/\tau$ ergibt sich aus der Koppelzeitkonstanten $\tau = RC$. Mit dieser als Multivibrator bezeichneten Schaltung werden Taktfolgen erzeugt.

10.2.4 Zählschaltungen

Von der Vielzahl der realisierten Zählschaltungen seien hier die in Fig. 10.23 gezeigten drei Ausführungsformen erläutert. Verbindet man in einer Reihe von Flip-Flop-Stufen den Takteingang des folgenden mit dem Q-Ausgang des vorhergehenden (s. Fig. 10.23a), so erhält

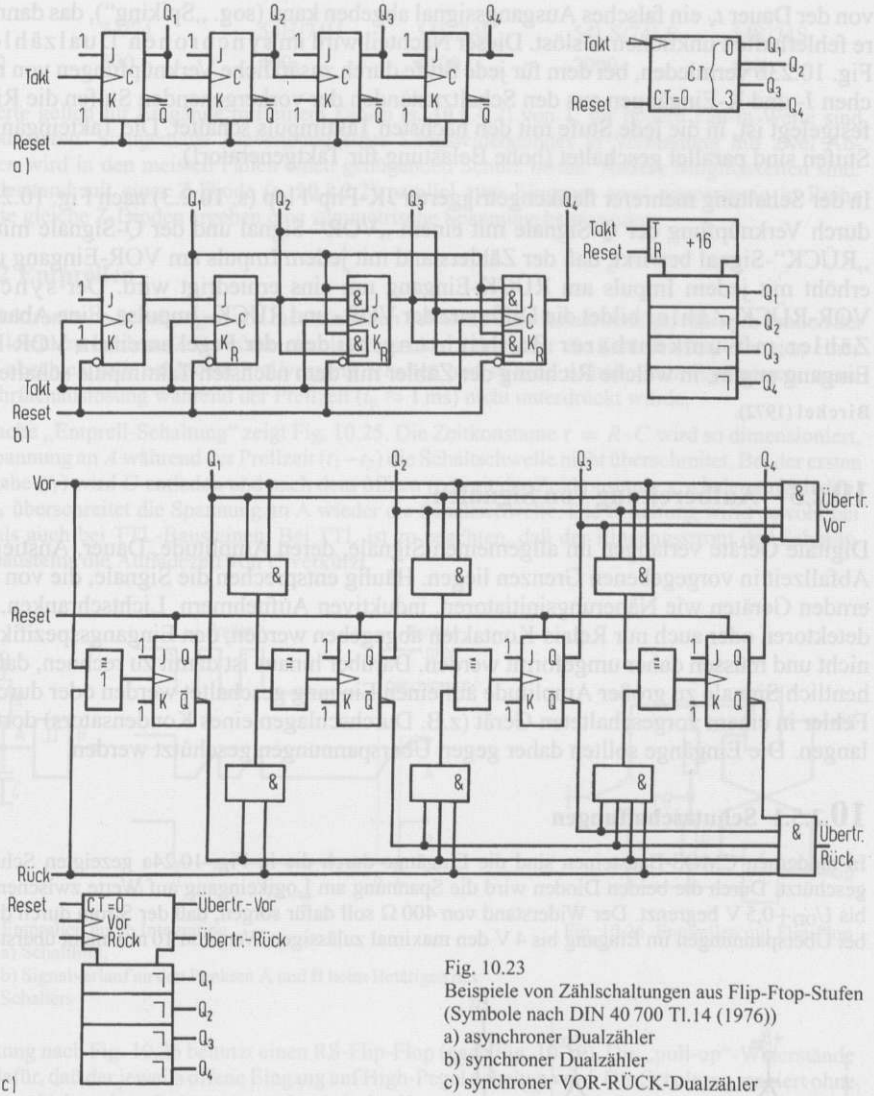


Fig. 10.23
Beispiele von Zählschaltungen aus Flip-Ftop-Stufen (Symbole nach DIN 40 700 Tl.14 (1976))
a) asynchroner Dualzähler
b) synchroner Dualzähler
c) synchroner VOR-RÜCK-Dualzähler

man einen asynchronen Dualzähler (binary ripple counter). Jede Zählstufe besteht aus einem JK-Master-Slave-FF (s. **10**.2.3) und untersetzt im Verhältnis 2:1. Bei einer Eingangsfrequenz f_e erhält man an der n-ten Stufe eine Frequenz $f_n = f_e/2^n$. Die registrierte Zahl ergibt sich nach Gl. (10.22), wenn den Zuständen $Q_i = H$ die Binärziffern $\alpha_i = 1$ zugeordnet werden. Da in jeder Stufe eine Verzögerungszeit t_v vergeht, bis nach der auslösenden Flanke des Taktsignals der Q-Ausgang seinen Pegel ändert (vgl. **10**.2.3), folgt die n-te Stufe erst um $n \cdot t_v$ verzögert. Dies ist bei der Anwendung als Zeitschalter zu beachten, weil demzufolge ein Dekoder, wenn zunächst noch nicht alle Stufen geschaltet haben, für eine sehr kurze Zeit, etwa

von der Dauer t_v ein falsches Ausgangssignal abgeben kann (sog. „Spiking"), das dann weitere fehlerhafte Funktionen auslöst. Dieser Nachteil wird im synchronen Dualzähler nach Fig. 10.23b vermieden, bei dem für jede Stufe durch zusätzliche Verknüpfungen von mehrfachen J- und K-Eingängen aus den Schaltzuständen der vorhergehenden Stufen die Richtung festgelegt ist, in die jede Stufe mit den nächsten Taktimpuls schaltet. Die Takteingänge aller Stufen sind parallel geschaltet (hohe Belastung für Taktgenerator!).

In der Schaltung mehrerer flankengetriggerter JK-Flip-Flop (s. **10**.2.3) nach Fig. 10.23c wird durch Verknüpfung der Q-Signale mit einem „VOR"-Signal und der $\overline{Q}$-Signale mit einem „RÜCK"-Signal bewirkt, daß der Zählerstand mit jedem Impuls am VOR-Eingang um eins erhöht mit jedem Impuls am RÜCK-Eingang um eins erniedrigt wird. Der synchrone VOR-RÜCK-Zähler bildet die Differenz der VOR- und RÜCK- Impulse. Eine Abart ist der Zähler mit umkehrbarer Zählrichtung, bei dem der Pegel an einem VOR-RÜCK-Eingang angibt, in welche Richtung der Zähler mit dem nächsten Taktimpuls schalten soll.

Birchel (1972).

10.2.5 Aufbereitung von Signalen

Digitale Geräte verlangen im allgemeinen Signale, deren Amplitude, Dauer, Anstiegs- und Abfallzeit in vorgegebenen Grenzen liegen. Häufig entsprechen die Signale, die von ansteuernden Geräten wie Näherungsinitiatoren, induktiven Aufnehmern, Lichtschranken, Photodetektoren oder auch nur Relais-Kontakten abgegeben werden, den Eingangsspezifikationen nicht und müssen daher umgeformt werden. Darüber hinaus ist damit zu rechnen, daß versehentlich Signale zu großer Amplitude auf einen Eingang geschaltet werden oder durch einen Fehler in einem vorgeschalteten Gerät (z.B. Durchschlagen eines Kondensators) dorthin gelangen. Die Eingänge sollten daher gegen Überspannungen geschützt werden.

10.2.5.1 Schutzschaltungen

In modernen CMOS-Bausteinen sind die Eingänge durch die in Fig. 10.24a gezeigten Schaltungen geschützt. Durch die beiden Dioden wird die Spannung am Logikeingang auf Werte zwischen $-0{,}5$ V bis $U_{DD}+0{,}5$ V begrenzt. Der Widerstand von 400 Ω soll dafür sorgen, daß der Strom durch die Diode bei Überspannungen im Eingang bis 4 V den maximal zulässigen Wert von 10 mA nicht übersteigt.

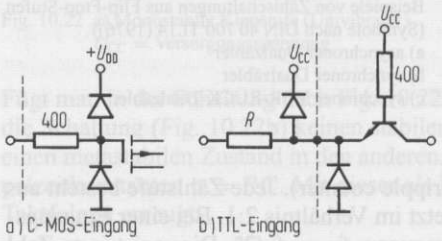


Fig. 10.24
Beispiele von Eingangs-Schutzschaltungen

Vor diesen Widerstand darf wegen des hochohmigen Eingangswiderstandes der Schaltung noch ein weiterer hochohmiger Serienwiderstand eingefügt werden (z.B. $R \geqslant 30$ kΩ, 2 W zum Schutz gegen die Netzspannung von 220 V). TTL-Schaltkreise sind nur durch eine Diode gegen negative Eingangsspannungen geschützt. Eine äußere Diode nach U_{CC} in Verbindung mit einem Längswiderstand R, (Fig. 10.24b) ergibt auch hier eine Spannungsbegrenzung. Der Widerstand darf folgende Werte nicht übersteigen:

Serie	54/74	54H/74H	54L/74L	54LS/74LS	54S/74S
R in Ω	500	400	4400	2000	400

Diese Werte gelten für Eingänge mit einem Fan-in (s. **10**.8.4.2) von 1, für höhere Fan-in-Werte sind sie zu reduzieren. Sinngemäße Anwendung eines Längswiderstandes in Verbindung mit zwei Abfangdioden wird in den meisten Fällen einen genügenden Schutz bieten. Andere Möglichkeiten sind: Längswiderstand mit einer Z-Diode (s. **10**.8.2.2) parallel zum Eingang; zwei gegensinnig in Reihe geschaltete gleiche Z-Dioden ergeben eine symmetrische Spannungsbegrenzung.

10.2.5.2 Entprellen

Werden elektronische Schaltungen von mechanischen Schaltern oder Relais betätigt, führt das Prellen der Kontakte leicht zu Fehlern. Z.B. würde ein elektronischer Zähler, der die Zahl der Schließungen eines Schalters abzählen soll, bei jedem Prellvorgang schalten und so eine zu hohe Zahl registrieren, wenn diese Mehrfachauslösung während der Prellzeit ($t_p \approx 1$ ms) nicht unterdrückt würde.

Eine einfache „Entprell-Schaltung" zeigt Fig. 10.25. Die Zeitkonstante $\tau = R \cdot C$ wird so dimensioniert, daß die Spannung an A während der Prellzeit ($t_1 - t_2$) die Schaltschwelle nicht überschreitet. Bei der ersten Kontaktgabe (t_1) wird C entladen und nach dem öffnen (t_3) mit der Zeitkonstante τ wieder aufgeladen. Erst bei t_4 überschreitet die Spannung an A wieder die Schaltschwelle. Die Schaltung wirkt sowohl bei CMOS- als auch bei TTL-Bausteinen. Bei TTL ist zu beachten, daß der Eingangsstrom des Schmitt-Trigger Bausteins die Aufladezeit von C verkürzt.

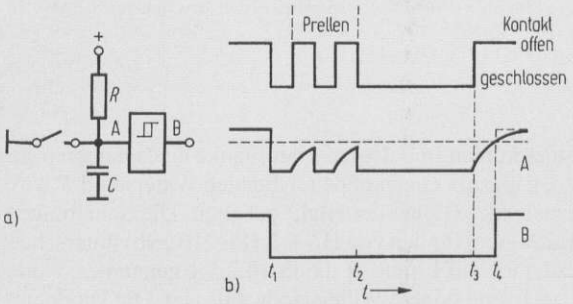


Fig. 10.25 Entprellen durch Integration
a) Schaltbild,
b) Signalverlauf an den Punkten A und B beim Betätigen des Schalters

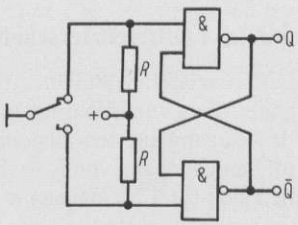


Fig. 10.26 Entprellen mit Flip-Flop

Die Schaltung nach Fig. 10.26 benutzt einen RS-Flip-Flop (nach Fig. 10.19). Die „pull-up"-Widerstände R sorgen dafür, daß der jeweils offene Eingang auf High-Pegel gehalten wird. Die Schaltung reagiert ohne Verzögerung. Neben dem Q-Ausgang steht auch das Komplement $\overline{Q}$ zur Verfügung. Die Schaltung ist für CMOS und TTL gleich gut zu verwenden. Der einzige Nachteil: Umschalter anstelle eines einfachen Schalters notwendig.

10.2.5.3 Schwellenschalter (Schmitt-Trigger)

Langsam veränderliche Spannungen am Eingang eines einfachen Gatters rufen, wegen der stets vorhandenen Störspannungen (Rauschen, Brummen), an dessen Ausgang Störimpulse hervor (s. Fig. 10.27). Beim Schmitt-Trigger werden diese Schwierigkeiten vermieden, sofern die Störspannung kleiner als die Spannungsdifferenz zwischen oberer und unterer Schaltschwelle (Hysterese) ist. Beispiele: Ableitung eines 50 Hz-Taktes aus der Netzspannung

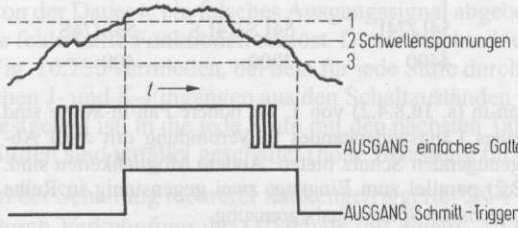


Fig. 10.27
Zum Verhalten einfacher Gatter und Schmitt-Trigger gegenüber zeitlich langsam veränderlichen, gestörten Eingangssignalen

oder Steuerung eines Zweipunktreglers durch temperaturproportionale Spannungen. Schmitt-Trigger sind als integrierte Schaltkreise erhältlich (TTL: 7413, 7414, 74 132; CMOS: 4093, 40 106). Bei CMOS-Schaltungen hängen die Schaltschwellen von der Betriebsspannung ab. Eine einstellbare Hysterese erhält man durch äußere Beschaltung eines Logik-Gatters mit zwei Widerständen R_G und R_E (s. Fig. 10.28a). Die Übertragungskennlinie $U_A = f(U_E)$ zeigt Fig. 10.28b.

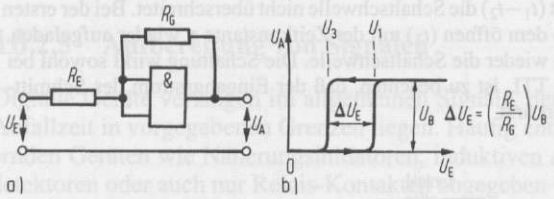


Fig. 10.28
a) Schmitt-Trigger aus UND-Gatter
b) Übertragungskennlinie, U_1 und U_3 entsprechen oberer bzw. unterer Schwelle in Fig. 10.27

10.2.5.4 Differenzierschaltungen

Differenzieren heißt hier: Ableitung eines kurzen Impulses aus der Flanke eines längeren Signals. Dazu wird das Eingangssignal U_E kapazitiv eingekoppelt. Über den Widerstand R wird der Eingang auf dem Pegel des Ruhezustandes (Low oder High) gehalten. Die Schaltungen für den Übergang von L → H (Fig. 10.29a) und für den von H → L (Fig. 10.29b) unterscheiden sich für TTL-Bausteine. Zum Festhalten auf L muß R die in **10**.2.5.1 genannten Werte unterschreiten. Beim Schalten von H → L sind höhere Widerstände zulässig. Die Diode unterdrückt das differenzierte Signal der unerwünschten Flanke. Fig. 10.29c zeigt die Schaltung für CMOS.

Zur Ableitung von Impulsen, deren Dauer von der Größenordnung der Verzögerungszeiten des einzelnen Gatters ist, wird häufig auch eine Schaltung nach Fig. 10.30 verwendet. Die gezeigte Ausführung liefert ein Signal für die Vor- und Rückflanke des Eingangsimpulses. Das wesentliche Prinzip erkennt man im unteren Zweig. Das Signal vom Ausgang von G_3 gelangt einmal direkt zum einen Eingang des NAND-Gatters G_4 und zum anderen über eine ungerade Anzahl n von Invertern, so daß die Und-Bedingung zwischen den Eingängen von G_4 nur für die Dauer der Verzögerung der Inverter erfüllt wird.

10.2.6 Signalausgänge

Häufig vorkommende Ausgangsschaltungen sind in Fig. 10.31 zusammengestellt. Bei der Ansteuerung von Relais (Fig. 10.31a) ist stets die beim Abschalten des Stromes durch die Relaisspule auftretende Spannungsspitze durch eine Diode abzufangen. Ströme

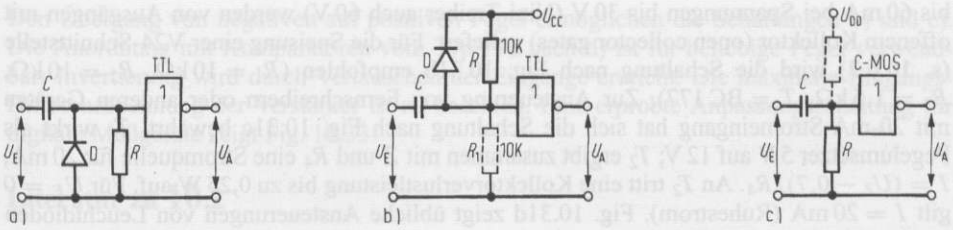


Fig. 10.29 Verschiedene Schaltungen zur Eingangsdifferentiation
Mit dem Widerstand R wird der Eingangspegel auf „High" (Versorgungsspannung, U_{CC} bzw. U_{DD}) oder „Low" (Masse) gehalten
a) Differentiation am Eingang eines TTL-Bausteines, Flanke Low → High
b) wie a), aber für Flanke High → Low. Mit R_1 kann die Schaltschwelle variiert werden
c) Eingangsdifferentiation für C-MOS-Schaltungen, Übergang Low → High, gestrichelte Anordnung von R für High → Low

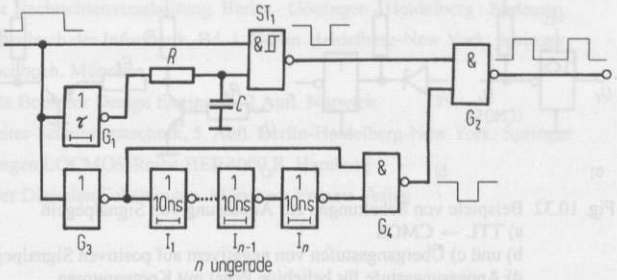
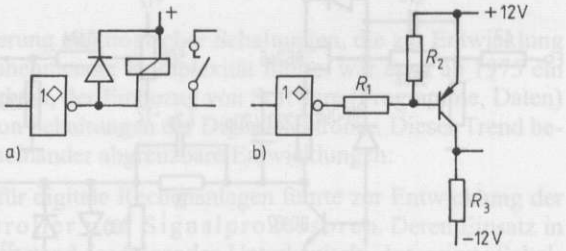


Fig. 10.30
Schaltung zur Differentiation mit Hilfe von Verzögerungsgliedern
Differentiation der Rückflanke: Unterer Zweig durch Laufzeit mehrerer Gatter
Differentiation der Vorderflanke: Oberer Zweig durch Umladen von C



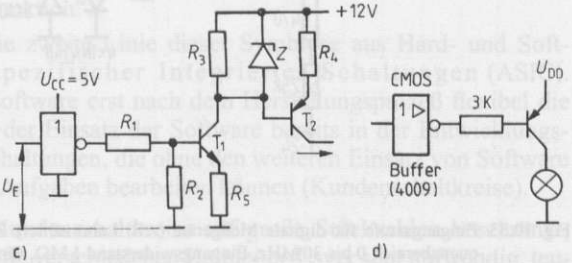


Fig. 10.31
Ausgangsschaltungen
a) Relaistreiber mit offenem Kollektorausgang,
b) Steuerung einer V 24 Schnittstelle,
c) Treiber für 20 mA-Fernschreiber,
U_E Eingangsspannung,
d) Lampen-Treiber, U_{CC} Versorgungsspannung der TTL-Bausteine

bis 60 mA bei Spannungen bis 30 V (Nixi-Treiber auch 60 V) werden von Ausgängen mit offenem Kollektor (open collector gates) geliefert. Für die Speisung einer V24-Schnittstelle (s. **10**.6.2) wird die Schaltung nach Fig. 10.31b empfohlen ($R_1 = 10\,\text{k}\Omega$, $R_2 = 10\,\text{k}\Omega$, $R_3 = 1{,}5\,\text{k}\Omega$, $T = \text{BC}\,177$). Zur Ansteuerung von Fernschreibern oder anderen Geräten mit 20-mA-Stromeingang hat sich die Schaltung nach Fig. 10.31c bewährt. T_1 wirkt als Pegelumsetzer 5 V auf 12 V; T_2 ergibt zusammen mit Z und R_4 eine Stromquelle für 20 mA, $I = (U_Z - 0{,}7)/R_4$. An T_2 tritt eine Kollektorverlustleistung bis zu 0,25 W auf. Für $U_E = 0$ gilt $I = 20\,\text{mA}$ (Ruhestrom). Fig. 10.31d zeigt übliche Ansteuerungen von Leuchtdioden und Anzeigelämpchen. Schaltungen zur Ansteuerung von Kabeln und Leitungen s. **10**.7.3.

Pegelumsetzer dienen der Kopplung von Geräten mit unterschiedlichen logischen Pegeln. Einige Beispiele zeigt Fig. 10.32. Der offene Kollektorausgang (open collector) a) ist stets bei positiver Polarität von Eingangs- und Ausgangspegel angezeigt (insbesondere TTL→CMOS). Für $U_E > U_{CC}$ ist eine Schutzschaltung nach. **10**.2.5.1 vorzusehen. Für den Übergang CMOS→TTL sind verschiedene Wandler im Handel (z.B. 4049/50).

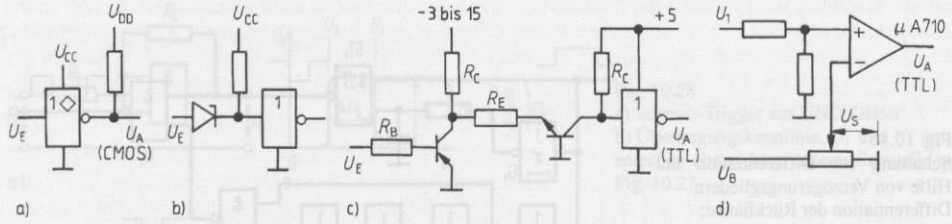


Fig. 10.32 Beispiele von Schaltungen zur Anpassung von Signalpegeln
a) TTL → CMOS
b) und c) Übergangsstufen von negativem auf positiven Signalpegel
d) Anpassungsstufe für beliebige Pegel mit Komparatoren

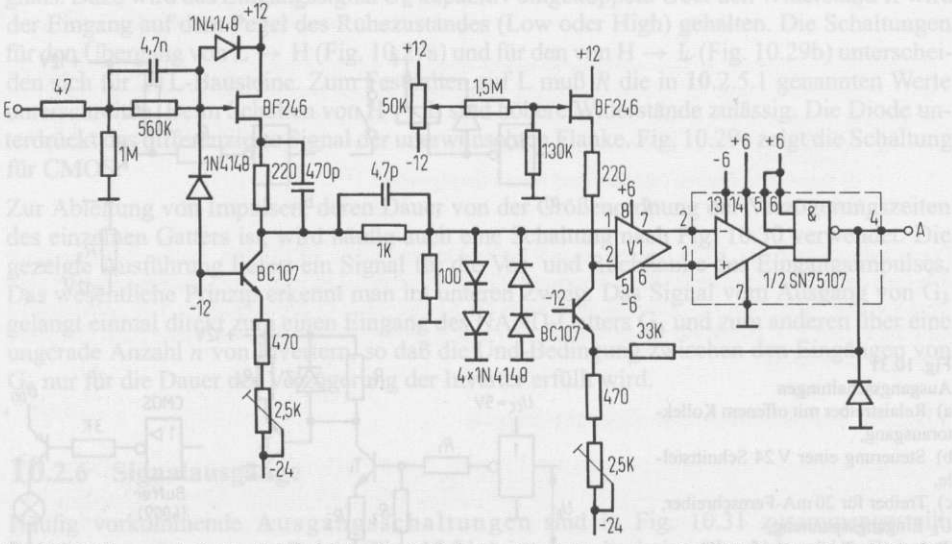


Fig. 10.33 Eingangsstufe für digitale Meßgeräte (z.B. Laborzähler) Eingangsempfindlichkeit 50 mV bis 10 V, Frequenzbereich 0 bis 30 MHz, Eingangswiderstand 1 MΩ, Eingangskapazität 30 pF.

Den Übergang von negativen auf positiven Pegel ermöglichen die Schaltungen b) und c). Die Anordnung mit Komparatoren vom Typ 710 nach d) ist für beliebige Pegel verwendbar; Invertierung wird durch Vertauschen der Eingänge erreicht. Die maximale Eingangs-Differenzspannung der Verstärker ist zu beachten. Eine erprobte Anpassungsschaltung für digitale Meßgeräte zeigt Fig. 10.33.

Literatur zu 10.2

Birchel, R. (1972): Elektronische Zähltechnik, Würzburg: Vogel Verlag

DIN 40 700, Teil 14 (1976): Schaltzeichen, Digitale Informationsverarbeitung. Berlin-Köln: Beuth (In der Literatur werden häufig noch die alten Schaltzeichen der Ausgabe von 1963 verwendet.)

Haack, O. (1984): Einführung in die Digitaltechnik. 4.Aufl. Stuttgart: Teubner

Heim, K. (1973): Schaltungsalgebra, 4.Aufl. Berlin-München: Siemens AG

Lancaster, D. (1980): Das CMOS-Kochbuch. Vaterstetten: I.W.T.

Leonhardt, E. (1976): Grundlagen der Digitaltechnik. München-Wien: Hanser

Meinke, H.; Gundlach, F. W. (1968): Taschenbuch für Hochfrequenz. Berlin-Heidelberg-New York: Springer

Steinbuch, K. (1962) : Taschenbuch der Nachrichtenverarbeitung. Berlin - Göttingen - Heidelberg : Springer

Steinbuch, K.; Weber, W. (1974): Taschenbuch der Informatik, Bd. 1. Berlin-Heidelberg-New York: Springer

Texas Instruments (1973): Das TTL-Kochbuch. München

Texas Instruments (1977): The TTL Data Book for Design Engineers, 2.Aufl. Norwich

Tietze, U.; Schenk, Ch. (1980): Halbleiter-Schaltungstechnik, 5. Aufl. Berlin-Heidelberg-New York: Springer

Valvo (1980): Integrierte Digitalschaltungen LOCMOS-Reihe HEF 4000 B. Hamburg

Zuiderveen, E. A. (1981): Handbuch der Digitalen Schaltungen. München: Franzis Verlag

10.3 Anwendungsspezifische Integrierte Schaltungen (R. Oppermann)

Neben der zunehmenden Miniaturisierung elektronischer Schaltungen, die zur Entwicklung Integrierter Schaltungen mit rasch zunehmender Komplexität führte, war etwa ab 1975 ein neuer revolutionärer Trend zu beobachten, der Einbezug von Software (Programme, Daten) in die Herstellung und Anwendung von Schaltungen der Digitalelektronik. Dieser Trend bewirkte in der Folge zwei deutlich voneinander abgrenzbare Entwicklungen:

1. Die Integration von Schaltungen für digitale Rechenanlagen führte zur Entwicklung der Mikroprozessoren, Mikrocontroller und Signalprozessoren. Deren Einsatz in Steuer- und Regelschaltungen ist Gegenstand des folgenden Unterkapitels „Integrierte Schaltungen mit softwaredeterminierter Funktion".

2. Nicht weniger bedeutend war die zweite Linie dieser Symbiose aus Hard- und Software, die Technik Anwendungsspezifischer Integrierter Schaltungen (ASIC). Während bei den Prozessoren die Software erst nach dem Herstellungsprozeß flexibel die Schaltungsfunktion bestimmt, führt der Einsatz der Software bereits in der Entwicklungsphase bei den ASICs zu Hardwareschaltungen, die ohne den weiteren Einsatz von Software unmittelbar anwendungsspezifische Aufgaben bearbeiten können (Kundenschaltkreise).

Zunächst war der Einsatz solcher Kundenschaltkreise auf große Stückzahlen beschränkt, weil die Herstellung der Masken für die verschiedenen Halbleiter-Layer sehr aufwendig, teu-

er und nur für hohe Stückzahlen lohnend war. Darüber hinaus dauerte die Fertigung, die ausschließlich beim Halbleiterhersteller stattfand, Monate. Auch heute noch ist der weitaus überwiegende Teil aller ASICs maskenprogrammiert und damit für den flexiblen Einsatz im Meßlaboratorium nicht geeignet.

Erst die Entwicklung Feldprogrammierbarer ASICs (Monolithic Memories 1978) erlaubte dem Anwender, nicht nur die Funktion der Schaltung mittels der Entwicklungssoftware festzulegen, sondern auch ihre „Herstellung" im Hause und ohne Zeitverlust mit einem Programmiergerät abzuschließen.

Feldprogrammierbare ASICs verknüpften die Vorteile einer Großserienfertigung (d. h. geringe Preise) mit der Möglichkeit, individuell der Aufgabe angepaßte digitale Laborschaltungen in Einzelstückzahlen herstellen zu können. Diese Bausteine werden heute unter einer Vielzahl von Namen angeboten. Hier werden nur wenige dieser Bezeichnungen benötigt, die signifikante Unterscheidungsmerkmale der einzelnen Typen charakterisieren.

10.3.1 Programmierbare Logikschaltungen (PLD)

10.3.1.1 Übersicht (Grundtypen und Technologie)

Der Begriff PLD (Programmable Logic Devices) wurde Oberbegriff für eine zweistufige Gatterlogik mit UND-Gattern in der ersten und ODER-Gattern in der zweiten Stufe sowie je einer Schaltmatrix vor den beiden Gatterblöcken. Mindestens eine davon ist programmierbar (Fig. 10.34). Es wird zwischen nur drei Grundtypen unterschieden:

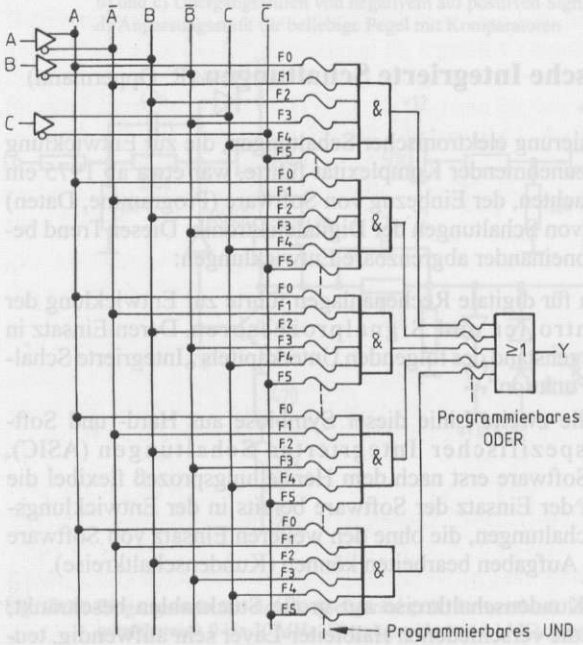


Fig. 10.34
Grundstruktur einer programmierbaren Logikschaltung mit Schmelzsicherungen (Fuses) vor UND- bzw. ODER-Gattern (Schimmer (1990))

– PAL (Programmable And Logic)
– PROM (Programmable Read Only Memory)
– PLA (Programmable Logic Array)

Die Einteilung der Grundtypen, die von der Entwicklungssoftware drei verschiedene Strategien erfordert (realisiert durch drei verschiedene Programme), bleibt auch dann dominant, wenn die Bausteine erheblich unterschiedliche Ausgangsschaltungen besitzen.

Man findet Bausteine mit

– fester Ausgangspolarität (positiv, negativ),
– programmierbarer Ausgangspolarität (X-PALs),
– Open-Kollektor-, Tristat-Ausgängen,
– abschaltbaren Ausgängen (Output-Enable),
– Ausgängen mit Registern für Zählschaltungen (Register-PLDs),
– Ausgängen, die durch Rückführungen in die UND-Gatter-Matrix gleichzeitig Eingang und Ausgang sind,
– oder in Verbindung mit Output-Enable wahlweise Eingang oder Ausgang sein können,
– sogenannten Makrozellen (siehe **10**.3.1.5), die die Programmierung aller oben genannten Varianten gestatten.

Eine Sonderstellung nehmen die Multiple-Array-Schaltungen ein, in denen bis zu 32 Makrozellen-PLDs in einem Chip durch ein programmierbares Leitungsnetz variabel miteinander verbunden werden können. Dies hat zur Folge, daß das dominante Merkmal einer Zweistufigkeit bei Multiple-Arrays nur im Bereich der einzelnen Makrozellen-Module anzutreffen ist. Auch in bezug auf die erreichbare Komplexität der Struktur heben sich Multiple-Arrays von den traditionellen PLDs ab. Sie besitzen durch die Verbindungsnetze eine extreme Zahl zusätzlicher Freiheitsgrade. Außerdem können Multiple-Arrays bis zu 400 Flip-Flops und mehrere tausend Gatterschaltungen haben. Der Einsatz dieser komplexen Schaltungen dürfte daher die Möglichkeiten eines Meßlaboratoriums überfordern.

Agraval (1991)

Bei der technologischen Realisierung von PLD-Schaltungen finden alle Varianten der Standardlogik Anwendung.

Man findet

– Bipolare Schaltungen (eine Versorgungsspannung, mittlere Stromaufnahme und Geschwindigkeit),
– CMOS-Schaltungen (eine Versorgungsspannung, sehr geringer Verbrauch, neue Entwicklungen sehr schnell)
– Z-PALs, die durch zusätzliche Power-Down-Schaltungen im Stillstand nahezu ohne Stromaufnahme sind,
– ECL-Schaltungen mit höchster Geschwindigkeit und Stromaufnahme, häufig mit mehreren Versorgungsspannungen.

Die programmierbaren Schaltmatrizen bestehen bei den PLDs der ersten Generation aus Sicherungsdrähten, die zunächst alle Eingänge mit allen Ausgängen verbinden und bei der Programmierung bis auf einen Rest, der die gewünschte Schaltung realisiert, zerstört werden (Fig. 10.35). Alle neueren PLDs haben dagegen Transistoren an den Matrix-Punkten, die bei der Programmierung die Verbindung herstellen (Antifuses). Da solche Verbindungen reversibel gestaltet werden können, gibt es seit einigen Jahren elektrisch oder UV-löschbare

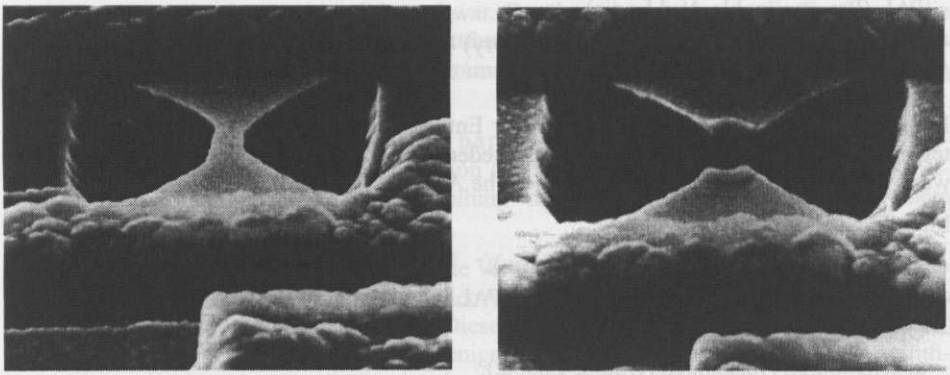


Fig. 10.35 REM-Foto eines Schmelzpfades bei etwa 15 000-facher Vergrößerung im intakten Zustand (a) und nach der Programmierung (b) (Werkbild AMD)

PLDs mit der Möglichkeit einer Reprogrammierung (GALs und EPLDs). Petersen (1988); Mehlo (1989)

10.3.1.2 Grundlagen der PLD-Technik

Die elementare PLD-Zelle Für das einfachste Grundelement einer PLD-Schaltung mit zwei Eingängen I_1 und I_2 und einem Ausgang S (Fig. 10.36a) kann man als allgemeine Gleichung für das Ausgangangssignal formulieren

$$S = (I_1 * F_1) * (/I_1 * F_2) * (I_2 * F_3) * (/I_2 * F_4) + (I_1 * F_5) * (/I_1 * F_6) * (I_2 * F_7) * (/I_2 * F_8) \quad (10.42)$$

Dabei sind F_1 bis F_8 die Schmelzpfade (Fuses) der UND-Matrix, die im unprogrammierten Zustand alle noch intakt sind ($F_i = 1$). Zu einer sinnvollen logischen Grundschaltung programmiert wird das Grundelement, indem einige Pfade zerstört werden.

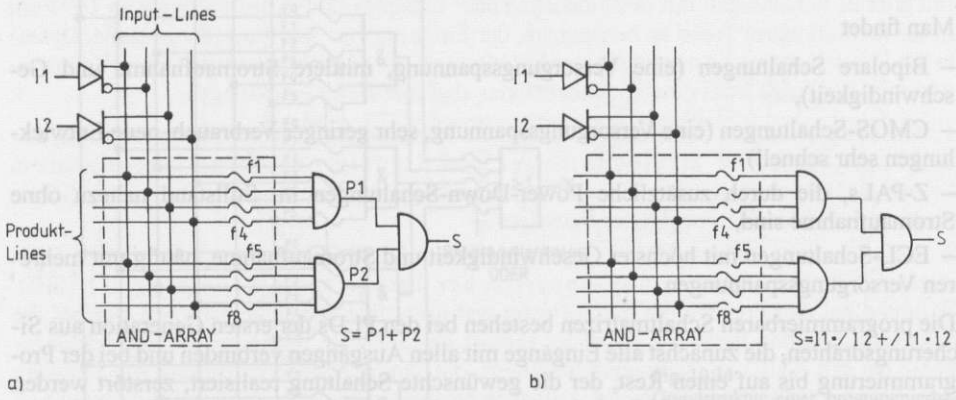


Fig. 10.36 a) Schaltung einer PAL-Zelle im Anlieferungszustand. Alle Sicherungspfade sind noch intakt
b) Programmierung einer sinnvollen Schaltung (EXOR-Gatter) durch Zerstörung einiger Schmelzsicherungen

Für die so entstandene neue Schaltung (Fig. 10.36b) gilt bei Substition mit den Werten „1" für erhalten gebliebene und „0" für offene Sicherungspfade

$$S = I_1 * /I_2 + /I_1 * I_2 \tag{10.43}$$

Aus der unprogrammierten Grundschaltung ist ein Exklusiv-ODER-Gatter (XOR) geworden.

Da jedes der beiden UND-Gatter aus dem echten und dem invertierten Eingangssignal bereits alle vier elementaren Grundverknüpfungen realisieren kann (Fig. 10.37), kann durch Anwendung der de Morganschen Regeln (s. **10**.2.2) einfach abgeleitet werden, daß die oben dargestellte Zelle die Realisation jeder möglichen kombinatorischen Verknüpfung von den beiden logischen Eingangsvariablen I_1 und I_2 ermöglicht, unter Einschluß solcher mit Inversion des Ausgangssignals. Diese Ableitung läßt sich in der Theorie auf PLDs mit n Eingängen und m Ausgängen erweitern.

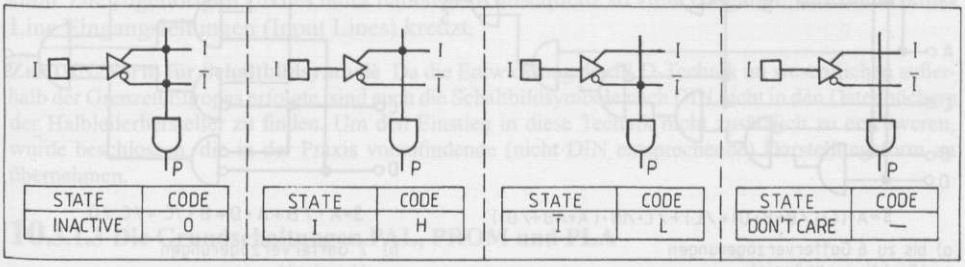


Fig. 10.37 Die Eingangsstufe mit den UND-Gattern in der ersten Stufe gestattet die Programmierung der vier elementaren Grundfunktionen von einer Eingangsvariablen. Eine anschließende ODER-Verknüpfung der Ausgangssignale P mehrerer Zellen erlaubt jede mögliche logische Verknüpfung der beteiligten Eingangsvariablen (Quelle: Valvo)

An jedem Ausgang der UND-Gatter in der Eingangsstufe entstehen Terme der Form

$$P_i = (I_1 * F_1) * (/I_1 * F_2) * \cdots * (I_n * F_{2n-1}) * (/I_n * F_{2n}) \tag{10.44}$$

(sog. Produktterme wegen des Verknüpfungszeichens)

Diese Produktterme P_i werden für jeden der m Ausgänge durch die ODER-Gatter der zweiten Stufe verknüpft zu Produktsummen

$$S_k = P_1 + P_2 + P_3 + \cdots + P_{2^n-1} + P_{2^n} \tag{10.45}$$

Da es zur Hälfte der möglichen Produktterme einen zweiten Produktterm gibt, der in einer ODER-Verknüpfung das triviale Ergebnis 1 liefert, ist die zur sinnvollen Verknüpfung von n Eingangsvariablen erforderliche Zahl von Eingängen der ODER-Gatter bei PLDs mit fester Ausgangspolarität

$$P_{\text{erf 1}} = \frac{2^n}{2} \tag{10.46a}$$

Bei PLDs mit programmierbarer Ausgangspolarität kann dagegen die Hälfte der möglichen Funktionen aus der einfachen Inversion der anderen Hälfte gewonnen werden. Dieser zusätzliche Freiheitsgrad führt zu

$$P_{\text{erf}\,2} = \frac{2^n}{4} \tag{10.46b}$$

Für die Praxis lassen sich aus dieser Betrachtung zwei Regeln ableiten:

1. Mehrstufige Schaltnetze beliebiger Stufenzahl. (z. B. Fig. 10.38a) werden bei ihrer Umsetzung in eine PLD-Schaltung grundsätzlich durch eine Gatterschaltung mit nur zwei Stufen realisiert (Fig. 10.38b).

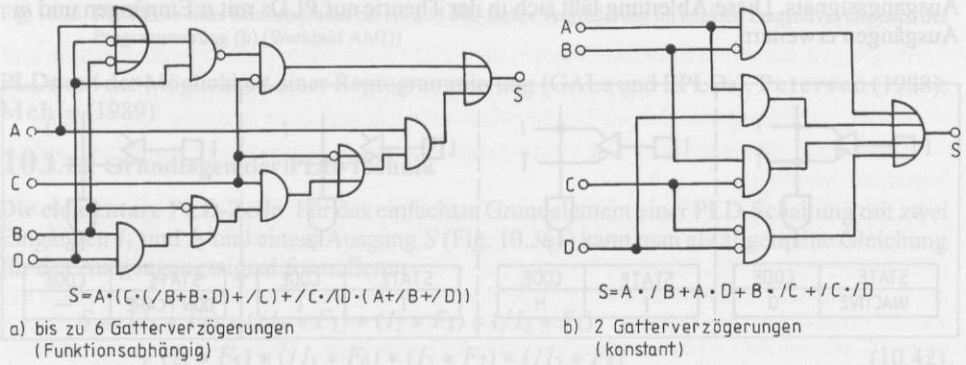


Fig. 10.38 Mehrstufige Schaltnetze werden als PLD-Schaltung grundsätzlich zweistufig realisiert, mit erheblichen technischen Vorteilen

Das bedeutet, daß bei PLD-Schaltungen zwischen Eingängen und Ausgängen grundsätzlich nur zwei Gatterverzögerungszeiten auftreten, und zwar für alle Ausgänge gleich, ein erheblicher Vorteil gegenüber Schaltungen, die mit Bausteinen der Standard-Logikfamilien realisiert sind!

2. Die Umsetzung einer Digitalschaltung in ein PLD kann durch die Zahl der Ein- und Ausgänge des gewählten Bausteins und die Zahl möglicher Produktterme erschwert werden.

Viele PLDs aus den ersten Entwicklungsjahren haben maximal 8 Produktterme. Nach Gl. (10.46) können damit nur 4 bzw. 5 Eingangsvariable restriktionsfrei verknüpft werden. Deshalb müssen bei der Entwicklung der Logik für die Funktion nicht unbedingt relevante Eingangsvariable aus dem Datensatz einer PLD-Schaltung eliminiert werden oder es muß versucht werden, durch einen Wechsel der Ausgangspolarität eine geringere Zahl von Produkttermen zu erreichen.

Form der Darstellung (PLD-Symbolik) Die für die elementare PLD-Zelle gewählte ausführliche Darstellung aller Leitungen einer Schaltung ist für die umfassenden Strukturen, wie sie in den angebotenen PLD-Bausteinen tatsächlich auftreten, zu detailliert und verwirrend.

Es hat sich daher in dieser Technik schon früh eine einfachere und verdichtete, von der üblichen Norm abweichende, aber dennoch eindeutige Symbolik durchgesetzt (Fig. 10.39).

Wesentlichster Gegenstand der Vereinfachung ist die Zusammenfassung aller Eingangsleitungen der UND-Gatter in der Eingangsstufe zu einer einzigen Linie (Produkt Line). Die Produkt

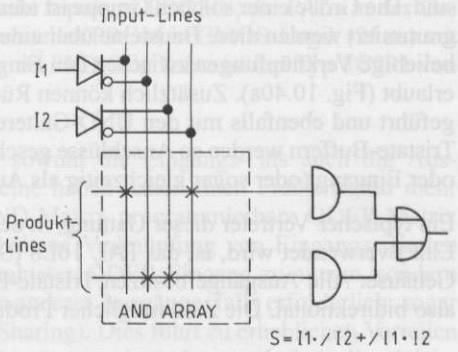


Fig. 10.39
Darstellung der PAL-Zelle nach Fig. 10.36b in der PLD-Symbolik. Jede Produkt Line hat soviel Leitungen zu den Eingängen des UND-Gatters, wie Eingangsleitungen gekreuzt werden. Die PLD-Symbolik bedeutet eine Vereinfachung der Darstellung ohne Verlust an Eindeutigkeit

Line symbolisiert eine vieladrige Leitung (wie in einem Bussystem), die mit allen echten und invertierten Eingangsleitungen in der Schaltmatrix über Sicherungspfade Verbindung haben kann. Die zugehörigen UND-Gatter haben als Konsequenz so viele Eingänge, wie die Produkt Line Eingangsleitungen (Input Lines) kreuzt.

Zur DIN-Norm für Schaltbildsymbole Da die Entwicklung der PLD-Technik im wesentlichen außerhalb der Grenzen Europas erfolgte, sind auch die Schaltbildsymbole nach DIN nicht in den Datenbüchern der Halbleiterhersteller zu finden. Um den Einstieg in diese Technik nicht zusätzlich zu erschweren, wurde beschlossen, die in der Praxis vorzufindende (nicht DIN entsprechende) Darstellungsform zu übernehmen.

10.3.1.3 Die Grundschaltungen PAL, PROM und PLA

PAL-Schaltkreise PALs stellen die älteste und immer noch am weitesten verbreitete Art von PLD-Bausteinen dar. Sie werden von vielen Herstellern angeboten. Ihre Schaltung besteht aus UND-Gattern in der ersten Stufe, deren Ausgänge über eine festverdrahtete Matrix in Gruppen zusammengefaßt mit den Eingängen der ODER-Gatter in der Ausgangsstufe verbunden

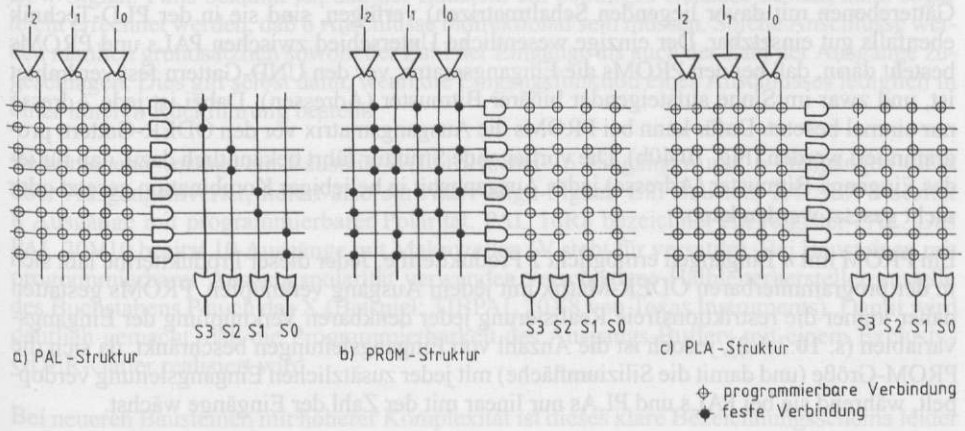


Fig. 10.40 Die drei PLD-Grundtypen PAL, PROM und PLA. Die PLA-Schaltung erlaubt durch zwei programmierbare Matrizen eine größere Entwurfsfreiheit als die beiden anderen Grundtypen

sind. Die Größe einer solchen Gruppe ist identisch mit der Zahl möglicher Produktterme. Programmiert werden diese Bausteine über eine Matrix vor den Eingängen der UND-Gatter, die beliebige Verknüpfungen zwischen den Eingangssignalen (in wahrer oder invertierter Form) erlaubt (Fig. 10.40a). Zusätzlich können Rückführungen der Ausgangssignale in die Matrix geführt und ebenfalls mit den UND-Gattereingängen verknüpfbar sein. In Verbindung mit Tristate-Buffern werden so Anschlüsse geschaffen, die je nach Programmierung als Ausgang oder Eingang (oder sogar gleichzeitig als Aus- und Eingang) verwendet werden können.

Ein typischer Vertreter dieser Gattung, in dem die ursprüngliche Sicherungs-Technik (Fuse-Link) verwendet wird, ist das PAL 16L8 (Schaltschema s. Fig. 10.41) im 20-poligen DIL-Gehäuse. Alle Ausgänge besitzen Tristate-Buffer, 6 Ausgänge haben Rückführungen, sind also bidirektional. Die Zahl möglicher Produktterme ist für alle Ausgänge 7.

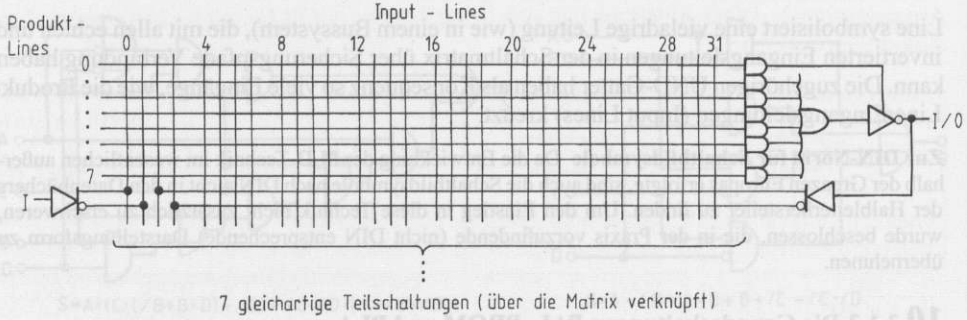


Fig. 10.41 Das klassische PAL 16L8 aus der ersten Generation. Dargestellt ist die Teilschaltung für einen der 8 Ausgänge (nach Unterlagen Texas Instruments)

PROM-Schaltkreise PROMs sind ursprünglich für die Datenspeicherung in Mikrocomputersystemen entwickelt worden und werden im überwiegenden Umfang nach wie vor in diesem Bereich eingesetzt. Da PROMs jedoch auch über die typische PLD-Struktur (zwei Gatterebenen mit davor liegenden Schaltmatrizen) verfügen, sind sie in der PLD-Technik ebenfalls gut einsetzbar. Der einzige wesentliche Unterschied zwischen PALs und PROMs besteht darin, daß bei den PROMs die Eingangsmatrix vor den UND-Gattern fest verdrahtet ist, und zwar im Sinne aufsteigender binärer Bitmuster (Adressen). Dabei ist jede Adresse nur einmal besetzt. Dafür kann bei PROMs die Ausgangsmatrix vor den ODER-Gattern programmiert werden (Fig. 10.40b). Die vorliegende Struktur führt bekanntlich dazu, daß für jedes Eingangs-Bitmuster (Adresse) jedes Ausgangsbit in beliebiger Kombination gesetzt oder nicht gesetzt werden kann.

Ein PROM mit n Eingängen ermöglicht 2 Produktterme. Jeder dieser Produktterme läßt sich in der programmierbaren ODER-Matrix mit jedem Ausgang verknüpfen. PROMs gestatten daher immer die restriktionsfreie Realisierung jeder denkbaren Verknüpfung der Eingangsvariablen (s. **10**.3.1.2), jedoch ist die Anzahl von Eingangsleitungen beschränkt, da sich die PROM-Größe (und damit die Siliziumfläche) mit jeder zusätzlichen Eingangsleitung verdoppelt, während sie bei PALs und PLAs nur linear mit der Zahl der Eingänge wächst.

EPROMS als bekannteste Vertreter dieser Bauteilgattung gestatten wegen ihrer erheblichen Zugriffszeit (Größenordnung 200 ns) als PLDs im allgemeinen nur den Einsatz in lang-

samen Schaltnetzen. Es gibt aber eine ganze Reihe bipolarer PROMS, die PALs in der Geschwindigkeit nicht nachstehen. Ein typisches schnelles PROM in CMOS-Logik (25 ns) mit 11 Eingängen und 8 Ausgängen für den Einsatz als PLD-Schaltnetz ist der Typ 8C291 von Cypress.

PLA-Schaltkreise Bei PLA-Schaltkreisen ist sowohl die Eingangs- als auch die Ausgangsmatrix frei programmierbar, diese Bausteine haben daher einen Freiheitsgrad mehr als PALs (Fig. 10.40c). Die zusätzlich zur UND-Matrix programmierbare ODER-Matrix erlaubt, die in einem UND-Gatterausgang enthaltene Verknüpfung von Eingangssignalen (den Produktterm) nicht nur an einem festverdrahteten ODER-Eingang zu nutzen, sondern gleichzeitig noch an weiteren ODER-Eingängen anderer Ausgänge, falls erforderlich, sogar an allen Ausgängen (sogenanntes Produktterm-Sharing). Dies führt zu erheblichen Vorteilen bei der Realisierung von Schaltungen, deren Ausgangsverknüpfung mehrfach die gleiche Eingangsverknüpfung enthält.

Ein gutes Beispiel für solche Applikationen sind Digital-Komperatoren. Bei einer PAL-Lösung wird bereits bei geringen Wortbreiten die Bausteinkapazität überschritten, weil mit jedem zusätzlichen Bit der Eingangsworte zwei zusätzliche Eingangsvariable hinzukommen. Vor allem müssen aber in jedem weiteren Produktterm alle bereits vorher berücksichtigten Eingangsvariablen erneut in eine Verknüpfung einbezogen werden. Bei einer PLA-Lösung ist es dagegen möglich, den in der vorangehenden Bitstelle gebildeten Produktterm noch einmal zu nutzen. Er wird lediglich mit zwei zusätzlichen Eingangsvariablen verknüpft.

Bedeutendster Hersteller von PLA-Bausteinen ist die Firma Signetics, die ihre Produkte mit IFL (Integrierte Fuse Logic) bezeichnet.

Nomenklatur der Standard-PALs Bei den PAL-Bausteinen der ersten Generation hat sich eine konsequente Bezeichnungsform entwickelt, die es erlaubt, die wichtigsten Eigenschaften aus dem Bauteilnamen abzuleiten. Die Bezeichnung PAL 16L8 ermöglicht die Deutung, daß es sich bei dem betreffenden Baustein um eine PAL-Struktur mit 16 Eingängen und 8 Ausgängen handelt. Die Ausgangs-Buffer haben Inverter, liefern also unmittelbar ein Aktiv-Low-Signal. Falls bekannt ist, daß der Baustein ein 20poliges Gehäuse besitzt, kann sogar leicht errechnet werden, daß 6 Anschlüsse bidirektional sein müssen. Solche Anschlüsse werden nämlich grundsätzlich sowohl der Zahl der Eingänge als auch der Zahl der Ausgänge zugeschlagen. Dies gilt selbst dann, wenn die Eingangsfunktion eines Anschlusses lediglich in einer inneren Rückführung besteht.

Entsprechend verfügt ein Baustein PAL 16H6 bei 16 Eingängen und 6 Ausgängen nicht über Ausgangsinverter, liefert also ein Aktiv-High-Signal. Ein Baustein PAL 10P8 besitzt 8 Ausgänge mit programmierbarer Polarität. PAL 16R8 bezeichnet ein Register-PAL. Das PAL 20V10 besitzt 10 Ausgänge mit Makrozellen (V steht für versatile). Bei Bausteinen mit programmierbarer Ausgangspolarität verwenden verschiedene Halbleiterhersteller anstelle des Buchstabens P auch das X (Beispiel: TIBPAL 20X8 von Texas Instruments). Damit wird deutlich gemacht, daß die Programmierbarkeit des Ausgangs-Buffers mit einem Exclusiv-ODER-Gatter realisiert wird.

Bei neueren Bausteinen mit höherer Komplexität ist dieses klare Bezeichnungsschema leider durchbrochen worden. Es erlaubt wohl auch bei den vielen Besonderheiten moderner PLD-Bausteine nicht genügend Differenzierungsmöglichkeiten.

10.3.1.4 Register-PLDs

Für sequentielle Anwendungen (Zähler, Schieberegister, Schrittsteuerungen) gibt es die drei PLD-Grundschaltungen auch mit Registern (Flip-Flop-Schaltungen) an den Ausgängen.

Register-PALs Der Baustein PAL 16R8, den viele Hersteller angebieten, kann für die Gattung der Register-PALs als besonders typisches Beispiel gelten (Fig. 10.42). Die Struk-

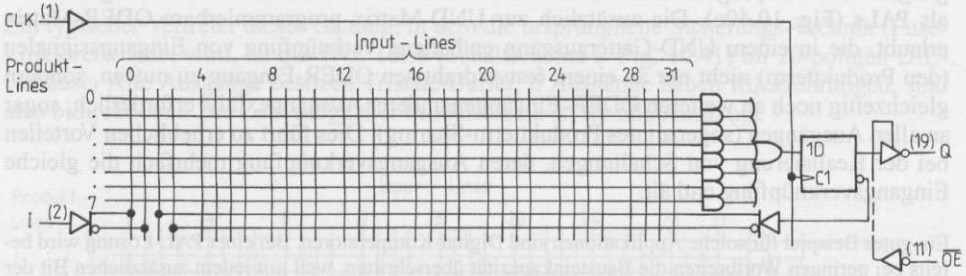


Fig. 10.42 Das Register-PAL 16R8 aus der Familie des 16L8. Dargestellt ist wieder eine der 8 Teilschaltungen (nach Unterlagen Texas Instruments)

tur dieses Bausteins zeigt, sieht man von den zusätzlich vorhandenen Registern ab, weitgehende Übereinstimmungen mit dem PAL 16L8. Unterschiedlich ist, daß der PAL 16R8 Rückführungen und extern gesteuerte Tristate-Buffer an allen 8 Ausgängen besitzt. Obwohl die 8 Anschlüsse nicht als reiner Eingang genutzt werden können, werden auch sie nach der Nomenklatur zur Zahl der Eingänge hinzugerechnet. Die externe Steuerung der Tristate-Buffer hat den Vorteil, daß 8 Produktterme und nicht nur 7 wie beim PAL 16L8 zur Verfügung stehen.

Zu den hervorzuhebenden typischen Eigenschaften des 16R8, die der weitaus überwiegende Teil aller Register-PALs aufzeigt, gehören

- die Verwendung von D-Flip-Flops,
- der synchrone (d. h. für alle Register gemeinsame) Takt,
- das Fehlen von SET- oder PRESET-Anschlüssen.

Die RESET-Kondition der Register wird bei PALs in aller Regel über einen allgemeinen Eingang und die Eingangsmatrix programmiert.

Prinzipiell können auf diesem Wege auch RS-Flip-Flops und JK-Flip-Flops verwirklicht werden. Dies kostet jedoch in der Regel wertvolle Produktterme, so daß Vorteile gegenüber Lösungen mit D-Flip-Flops nicht erzielt werden. T-Flip-Flops lassen sich dagegen in X-PALs mit Hilfe der zur Programmie-

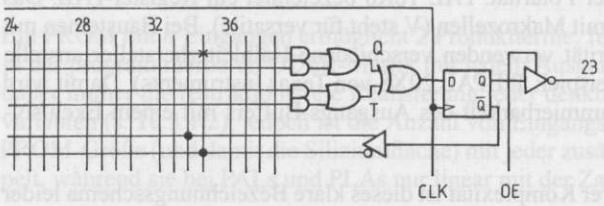


Fig. 10.43 X-PALs wie das 20X10 gestatten durch EXOR-Gatter nicht nur eine programmierbare Ausgangspolarität, sondern auch die Realisierung von T-Flip-Flops, die Zählschaltungen erheblich vereinfachen

rung der Ausgangspolarität vorgesehenen Exklusiv-ODER-Gatter realisieren, ohne die Zahl verfügbarer Produktterme einzuschränken (Fig. 10.43). Das ist für den Schaltungsentwickler sehr vorteilhaft, weil Zählerapplikationen mit T-Flip-Flops mit einer deutlich geringeren Zahl an Produkttermen zu verwirklichen sind als mit D-Flip-Flops.

Die synchrone Arbeitsweise sequentieller Logik-Schaltungen ist eine generelle Eigenschaft der PLDs, von der nur ganz wenige Bausteine abweichen. Sie hat gegenüber asynchronen Lösungen soviele Vorteile, daß auch Schaltwerke aus Bausteinen der Standard-Logikfamilien grundsätzlich synchron realisiert werden sollten, was immer möglich ist. Die beiden wichtigsten Eigenschaften der synchronen Digitaltechnik sind

– die über die gesamte Taktperiode eindeutige Größe des Ausgangsvektors (des von den Ausgangsvariablen gebildeten Bitmusters)

– die hohe Störsicherheit, die dadurch bewirkt wird, daß alle Flipflops gleichzeitig schalten und Störimpulse außerhalb des schmalen Zeitfensters keine Wirkung haben können.

Bei asynchrone Zählwerken ist dagegen ein erheblicher Aufwand erforderlich, um Schaltungen gegen ungewollte Kippvorgänge durch Einbruch der Versorgungsspannung und Mitnahmeeffekte (Übersprechen) immun zu machen. Darüber hinaus enthält der Ausgangsvektor asynchroner Zähler während der Übergangszeit von einem Zustand zum folgenden grundsätzlich falsche Werte, die zur Fehlfunktion führen können.

Register-PROMs Als typisches Register-PROM ist der Baustein CY8C245 von Cypress einzustufen. Er besitzt 9 Eingänge und 8 Tristate-Ausgänge mit externer Steuerung (Fig. 10.44).

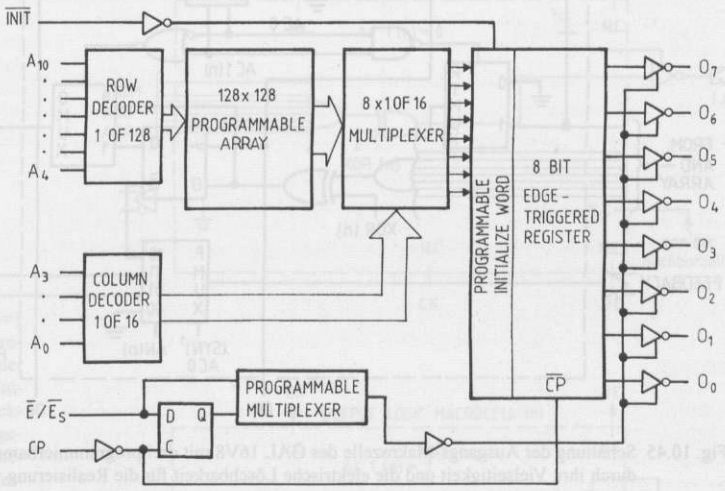


Fig. 10.44 Das Register-PROM CYC245 (Werkbild CYPRESS)

Register-PLAs Register-PLAs, von denen es nur wenige Typen von den Herstellerfirmen Valvo und Signetics gibt, erlauben durch JK-Flip-Flops mit SET- und RESET-Eingängen mehr Freiheitsgrade bei der Schaltungsentwicklung als Register-PALs, die nur über D-Flip-Flops verfügen. Als weitere Besonderheit finden wir bei ihnen neben den Ausgangsregistern sogenannte „vergrabene" Register, die nicht an den Ausgängen liegen. Diese zusätzlichen Register tragen der Tatsache Rechnung, daß in sequentiellen Schaltungen in der Regel nicht alle

Zustände, die eine Zählschaltung einnehmen kann, auch immer zu Ausgangssignalen führen müssen, also Bausteinanschlüsse verlangen. Die Hersteller bezeichnen diese Bausteine als Programmable Logic Sequenzer (PLS).

10.3.1.5 Makrozellen-PLDs

Das PAL 22V10 (AMD) war das erste Bauelement, in dem die sehr gute Idee der Ausgangs-Makrozelle zuerst verwirklicht wurde. Man versteht darunter komplexe Ausgangsschaltungen mit Registern und programmierbaren Multiplexern. Die Makrozelle gestattet für jeden Ausgang getrennt die Wahl

- zwischen der Eingangs- oder Ausgangsfunktion,
- der Ausgangspolarität,
- von kombinatorischen und Registerausgängen.

Die mit Abstand bekanntesten Vertreter dieser Gattung sind die Typen GAL 16V8 und GAL 20V8 (Lattice, SGS-Thompson und National Semiconductor).

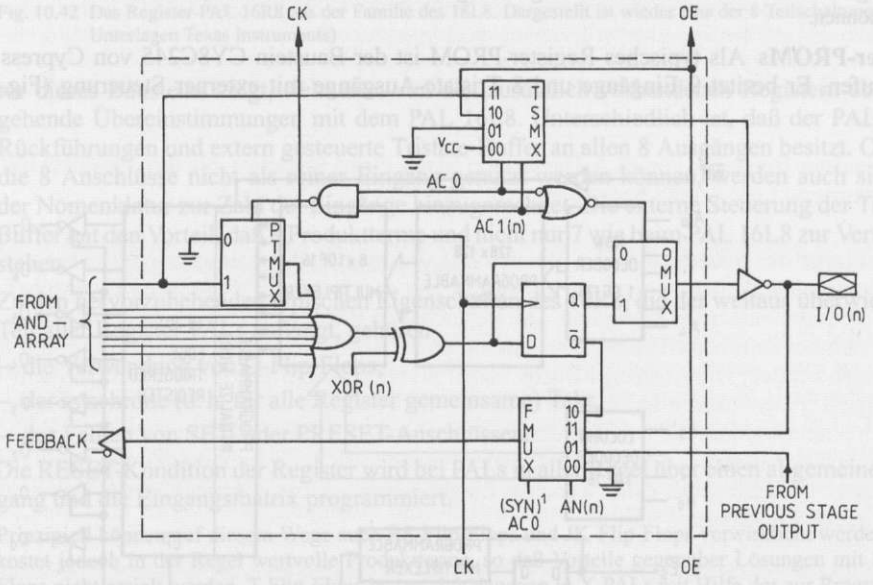
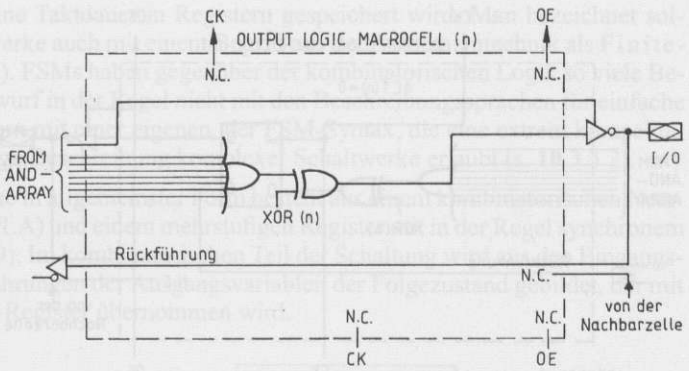


Fig. 10.45 Schaltung der Ausgangs-Makrozelle des GAL 16V8 mit drei programmierbaren Multiplexern. GALs sind durch ihre Vielseitigkeit und die elektrische Löschbarkeit für die Realisierung von Laborschaltungen hervorragend geeignet

GALs (Generic Array Logic) werden in einem speziellen CMOS-Prozeß hergestellt, der eine sehr hohe Geschwindigkeit mit elektrischer Löschbarkeit verbindet (E^2CMOS). Neue Versionen erreichen eine Verzögerungszeit von 7 ns (über 2 Gatter und die Multiplexer gemessen!).

Die beiden GALs lösen das Problem der Typenvielfalt auf elegante Weise. Sie ersetzen vollkommen etwa 50 verschiedene PALs der ersten Generation. Außerdem gestatten sie Konfigurationen, die es bisher nicht gab, beispielsweise die Realisierung eines einzigen Register-Ausgangs in einer kombinatorischen Schaltung.



XOR (n)	OUTPUT POLARITY
0	ACTIVE LOW
1	ACTIVE HIGH

Fig. 10.46
Der C7-Mode der Makrozelle für SYN=1, ACO=1 und AC1(n)=1. Alle 8 Ausgänge sind kombinatorische Ausgänge mit Rückführungen, die bei Abschaltung des aus der Matrix gesteuerten Tristate-Buffers auch als reine Eingänge benutzt werden können. Zwei weitere Anschlüsse (CK und OE) stehen als zusätzliche Eingänge zur Verfügung. Im C7-Mode besitzt jeder Ausgang 7 Produktterme

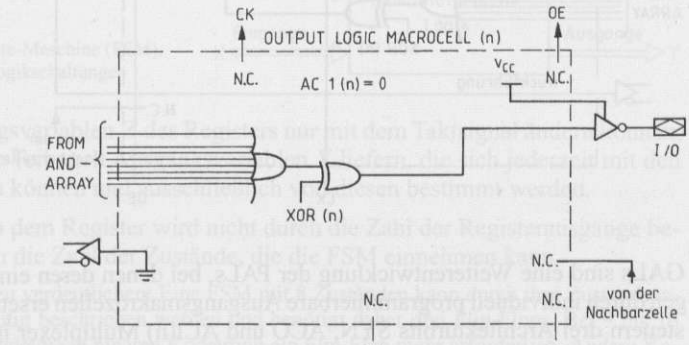
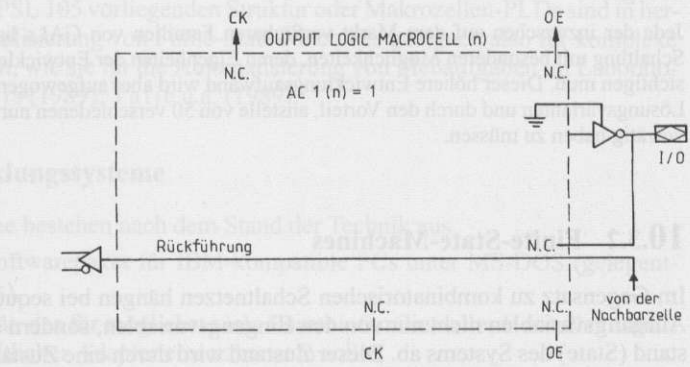


Fig. 10.47
Der C8-Mode für SYN=1 und ACO=0. AC1(n)=0 programmiert den Anschluß der Makrozelle zum kombinatorischen Ausgang ohne Rückführung, der immer eingeschaltet ist (keine Tristate-Funktion) und 8 Produktterme besitzt. Über AC1(n)=1 wird die betreffende Zelle durch die Nutzung ihres Rückführungsterms zum Eingang, der sein Signal vom Anschluß der außen liegenden Nachbarzelle erhält. Die beiden äußersten Zellen nutzen für die Eingangsfunktion die Anschlüsse CK und OE



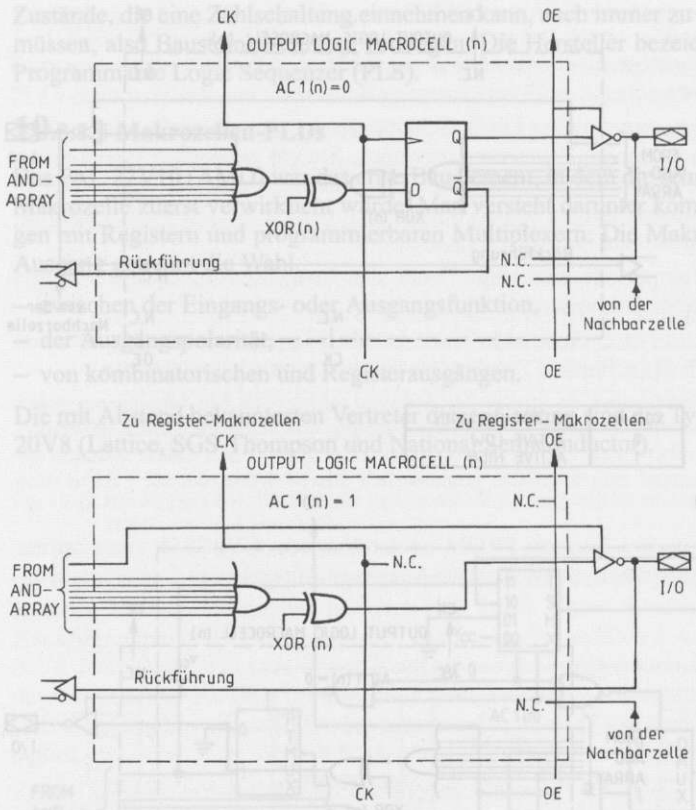


Fig. 10.48
Nur im R8-Mode für SYN=0 und ACO=1 können die Register genutzt werden. Anschluß CK (CLOCK) wird dabei für den synchronen Takt und Anschluß OE (OUTPUT ENABLE) für die zentrale Steuerung der Tristate-Buffer aller Register eingesetzt. AC1(n)=0 aktiviert die Registerfunktion der Zelle, AC1(n)=1 überbrückt das Register und bewirkt einen kombinatorischen Ausgang. Alle Ausgänge haben Rückführungen, die aber nur bei kombinatorischen Anschlüssen, bei denen die Tristate-Buffer individuell abgeschaltet werden können, als reiner Eingang nutzbar sind. Im R8-Mode haben Register-Ausgänge 8 und kombinatorische 7 Produktterme (alle Darstellungen des GAL 16V8 nach Werkunterlagen LATTICE)

GALs sind eine Weiterentwicklung der PALs, bei denen deren einfache Ausgangsschaltungen durch individuell programmierbare Ausgangsmakrozellen ersetzt sind (Fig. 10.45). Dabei steuern drei Architekturbits SYN, ACO und ACl(n) Multiplexer in der Makrozelle und bewirken dadurch drei grundverschiedene Modi C7, C8 und R8 des Bausteins (s. Fig.10.46 bis 10.48). SYN und ACO sind globale Steuerbits, ACl(n) dagegen auf den jeweiligen Ausgang bezogen.

Jede der inzwischen auf dem Markt verfügbaren Familien von GALs hat eine andere Makrozellen-Schaltung mit besonderen Möglichkeiten, deren Eigenheiten der Entwickler in seinem Entwurf berücksichtigen muß. Dieser höhere Entwicklungsaufwand wird aber aufgewogen durch die vielen möglichen Lösungsvarianten und durch den Vorteil, anstelle von 50 verschiedenen nur noch ein bzw. zwei Bauteile vorrätig haben zu müssen.

10.3.2 Finite-State-Machines

Im Gegensatz zu kombinatorischen Schaltnetzen hängen bei sequentiellen Schaltungen die Ausgangsvariablen nicht nur von den Eingangsvariablen, sondern auch vom jeweiligen Zustand (State) des Systems ab. Dieser Zustand wird durch eine Zustandsvariable repräsentiert,

deren Wert für jeweils eine Taktdauer in Registern gespeichert wird. Man bezeichnet solche sequentiellen Schaltwerke auch mit einem Begriff aus der Automatentechnik als Finite-State-Machines (FSM). FSMs haben gegenüber der kombinatorischen Logik so viele Besonderheiten, daß ihr Entwurf in der Regel nicht mit den Beschreibungssprachen für einfache Schaltnetze erfolgt, sondern mit einer eigenen, der FSM-Syntax, die eine extrem kompakte, vollständige und eindeutige Beschreibung komplexer Schaltwerke erlaubt (s. **10**.3.3.2).

Eine Finite-State-Maschine in allgemeinster Form besteht aus einem kombinatorischen Netzwerk (PAL, PROM oder PLA) und einem mehrstufigen Register mit in der Regel synchronem Takt und Reset (Fig. 10.49). Im kombinatorischen Teil der Schaltung wird aus den Eingangsvariablen und den Rückführungen der Ausgangsvariablen der Folgezustand gebildet, der mit dem nächsten Takt in das Register übernommen wird.

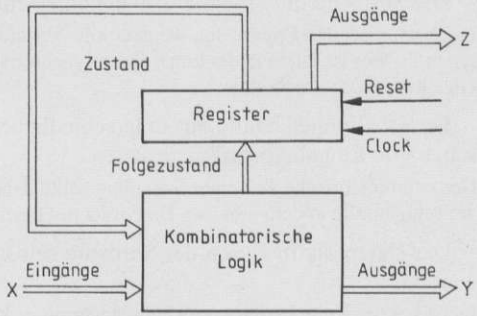


Fig. 10.49
Prinzipschaltung einer Finite-State-Maschine (FSM), Grundform aller sequentiellen Logikschaltungen

Während sich die Ausgangsvariablen Z des Registers nur mit dem Taktsignal ändern können, kann der kombinatorische Teil auch Ausgangsvariablen Y liefern, die sich jederzeit mit den Eingangsvariablen ändern können und ausschließlich von diesen bestimmt werden.

Die Zahl der Flip-Flops in dem Register wird nicht durch die Zahl der Registerausgänge bestimmt, sondern nur durch die Zahl der Zustände, die die FSM einnehmen kann.

Um das auf einfache Weise zu verdeutlichen: Eine FSM mit 8 Zuständen kann durch drei Zustandsvariable ($2^3 = 8$) binär eindeutig beschrieben werden und benötigt daher drei Flip-Flops. Bei 9 bis 16 Zuständen ist dagegen zur eindeutigen binären Kodierung ein Flip-Flop mehr erforderlich. Andere Kodierungen (1 aus N, Bi-Quinär, aber auch BCD bei mehr als 10 Zuständen) können mehr Flip-Flops erfordern.

Schaltungen mit der beim PSL 105 vorliegenden Struktur oder Makrozellen-PLDs sind in hervorragender Weise zur Realisierung von Finite-State-Machines geeignet, also für komplexe digitale Ablaufsteuerungen, wie sie für die Automatisierung von Meßaufgaben im Laboratorium benötigt werden. Reis (1989); Tatje u.a. (1989).

10.3.3 PLD-Entwicklungssysteme

PLD-Entwicklungssysteme bestehen nach dem Stand der Technik aus

– einem Entwicklungs-Softwarepaket für IBM-kompatible PCs unter MS-DOS (gelegentlich auch für Workstations),

– einem Programmiergerät, das für möglichst viele Bausteinfamilien aller Hersteller unmittelbar oder mit Hilfe erhältlicher Adapter einsetzbar sein sollte,

– und das über eine Standard-Rechnerschnittstelle (seriell oder parallel) bzw. über eine spezielle Schnittstelle mit zugehöriger Interfacekarte für den Rechner verfügen muß,
– dem Rechner, mit dem diese Aufgabe zusätzlich zu andereren Laboranwendungen bearbeitet wird.

10.3.3.1 Anforderungen an die Entwicklungssoftware

Eine geeignete Entwicklungssoftware muß die folgenden Eigenschaften haben:

– Das Eingabeformat muß eine Formulierung der Aufgabe auf Problemebene gestatten.
Ein Adreßdekoder sollte nach dieser Forderung durch Angabe der Adreßbereiche und des Verhaltens der Ausgangsvariablen vollständig beschrieben werden können.

– Die Software muß möglichst viele unterschiedliche Eingabeformate anbieten.
Es überfordert den Entwickler, wenn er alle Aufgabenstellungen in ein und demselben Format definieren muß. Wer ist schon in der Lage, einen synchronen Gray-Code-Vor-Rückwärts-Zähler mit Boolschen Gleichungen zu beschreiben?

– Da Schaltungen häufig aus unterschiedlicher Logik bestehen, sollte die Software das Mischen von Eingabeformaten gestatten.
Der kombinatorische Teil einer Schaltung sollte daher mit Boolschen Gleichungen zu beschreiben sein, der sequentielle in dem gleichen Datensatz mit einer geeigneteren Beschreibungsform.

– Der Datensatz muß von der Software selbsttätig auf Plausibilität und Konsistenz geprüft werden.
Es ist bei der Komplexität von Ablaufsteuerungen kaum möglich, als Entwickler einen von vornherein fehlerlosen Datensatz zu schreiben.

– Die bausteinbezogene Umsetzung des optimierten Ergebnisses muß der letzte Entwicklungsschritt sein. Vorher muß bereits eine Logiksynthese durch Simulation möglich sein. Damit scheiden nicht nur die sogenannten Fuse-Editoren aus, mit denen geplagte Entwickler in der Anfangszeit direkt die Nummern der zu zerstörenden Sicherungen in den Rechner oder sogar in das Programmiergerät eintippen mußten. Ungeeignet ist auch jede Software, die nur mit Mühe die Übertragung eines Datensatzes auf unterschiedliche Bausteine erlaubt.

– Das Ausgabeformat für den Programmierer muß dem internationalen JEDEC-Format (Joint Electron Device Council, USA) voll entsprechen.

– Die Software muß Testvektoren erzeugen, die den Test des fertigen Bausteins erlauben.

– Und schließlich muß die Software eine vollständige Dokumentation der Aufgabe und ihrer Lösung bieten.

Optimierungs-Algorithmen Damit eine prinzipiell mögliche Realisierung einer Schaltung in einem bestimmten Baustein durch das Auffinden ihrer Minimalform möglich wird, muß die Software

– über einen leistungsfähigen Optimierungs-Algorithmus verfügen, der in der Lage ist, die Minimalform einer Schaltung auch zu finden,
– bereits zu Beginn der Entwicklung die Struktur des gewählten Bausteins berücksichtigen können.

Die zweite Forderung steht im Widerspruch zum Wunsch, bausteinspezifische Daten erst nach einer Schaltungsoptimierung auf logischer Ebene in einem zweiten Entwicklungsschritt ein-

geben zu müssen. Leistungsfähige Entwicklungspakete lösen dieses Problem mit drei verschiedenen Compilern für die Grundtypen PAL, PROM und PLA.

Der PAL-Compiler hat zu berücksichtigen, daß bei PAL-Strukturen jedem Ausgang eine begrenzte Zahl von Produkttermen fest zugeordnet ist. Ziel der Optimierung ist daher, jede Funktion mit der absoluten Minimalzahl von Produkttermen zu realisieren. Da Produktterme nicht an anderen Ausgängen mitgenutzt werden können, muß jede Ausgangsfunktion gesondert optimiert werden.

Der PROM-Compiler kennt das Problem der Produkttermbeschränkung nicht. PROMs haben immer genug Produktterme, um jede mögliche Funktion der Eingangsvariablen realisieren zu können. Dagegen leiden PROMs unter der Restriktion, nicht genügend Ein- und Ausgänge zur Verfügung zu haben. Das Ziel der Optimierung ist daher die Entfernung überflüssiger Eingangs- und Ausgangsleitungen.

Der PLA-Compiler berücksichtigt, daß PLAs durch die ebenfalls programmierbare ODER-Matrix die Möglichkeit der Mehrfachnutzung von Produkttermen besitzen. Es ist daher Aufgabe des Minimierungs-Algorithmus, diese Mehrfachnutzung anzustreben. Das bedingt, daß Funktionen nicht mehr für jeden Ausgang gesondert optimiert werden, sondern für alle Ausgänge gleichzeitig (Bündeloptimierung).

Tatje u.a. (1989); Göritz (1989)

10.3.3.2 Bestandteile eines PLD-Eingabedatensatzes

Es gibt auf dem Markt neben herstellerabhängigen Softwareprodukten, die lediglich für wenige Bausteintypen eines Herstellers geeignet sind, eine Reihe von universellen Entwicklungspaketen für PLDs. Die bekanntesten sind

- LOG/iC von der Firma Isdata,
- ABEL von den Firmen FutureNet und Data I/O,
- CUPL von der Firma Logical Devices.

Der Verfasser setzt LOG/iC, das den gestellten Forderungen voll entspricht, seit einigen Jahren ein. Alle Beispiele in der Folge beziehen sich daher auf die Arbeit mit diesem Softwarepaket und die dabei gesammelten Erfahrungen.

Ein PLD-Datensatz für LOG/iC enthält drei wesentliche Teile:

1. Eine Beschreibung der Aufgabe auf logischer, bauteilunabhängiger Ebene,
2. hardwarespezifische Angaben (Typ des Bausteins, u.U. besondere Hinweise zu seiner Verwendung, Zuordnung der Eingangs- und Ausgangsvariablen zu bestimmten Anschlüssen),
3. Anweisungen an den Compiler über die gewünschten Ausgabedateien (JEDEC-Programmierfile, Dokumentation, Testvektoren etc.).

Für die Entwicklung auf logischer Ebene kann anstelle eines realen Bausteins ein fiktiver (HYPERPAL, -PLA, -PROM) eingesetzt werden. HYPER-PLDs haben keine Restriktionen, d.h. genügend Anschlüsse und Produktterme. Man erreicht auf logischer Ebene daher immer eine Lösung. LOG/iC bietet sogar über eine integrierte PLD-Bibliothek (Option) geeignete Bausteine für die Realisierung an. Ist eine Lösung mit einem einzigen Baustein nicht möglich, so wird auch die Aufteilung auf mehrere PLDs (Partionierung) selbsttätig durchgeführt.

Eingabe über Boolsche Gleichungen Auch sequentielle Schaltungen lassen sich (obwohl nicht üblich) durch Boolsche Gleichungen beschreiben. So gelten für die ersten beiden Stufen eines einfachen synchronen Binärzählers mit RESET und D-Flipflops die Gleichungen

$$QQ1 := /QQ1 \,\&\, /RESET \tag{10.47}$$

$$QQ2 := /QQ2\&QQ1 \,\&\, /RESET + QQ2 \,\&\, /QQ1\&/RESET \tag{10.48}$$

Dabei steht „&" in der LOG/iC-Syntax für UND, „+" für ODER, „#" für EXOR und „/" für die Negation. „:=" bedeutet, daß die links stehende Ausgangsvariable nach dem Takt den Wert übernehmen wird, der sich aus der rechts stehenden Funktion der Eingangsvariablen vor dem Takt ergibt. QQ1 und QQ2 treten sowohl als Eingangs- als auch als Ausgangsvariable auf. Dies ist mit der internen Rückführung dieser Signale zu erklären.

Für RES = 0 (kein Reset) wechselt QQ1 mit jedem Takt von 0 auf 1, QQ2 dagegen nur, wenn QQ1 = 1 ist, d.h. mit jedem zweiten Takt. Für RES = 0 (Reset) nehmen beide Ausgänge den Wert 0 ein.

Mit zunehmender Stufenzahl werden die Gleichungen rasch zu umfangreich und komplex, deshalb wird diese Form der Eingabe in der Regel für kombinatorische Schaltungen verwendet oder für sequentielle Schaltungen mit nur wenigen Stufen.

Tab. 10.2 Datensatz für einen Adreßdekoder mit Boolschen Gleichungen als Eingabeformat. Durch den Einsatz eines fiktiven Bausteins HYPERPLD kann die Entwicklung zunächst auf logischer Ebene durchgeführt werden

```
*IDENTIFICATION
 DEKODER FÜR MIKROCOMPUTERSYSTEM
 LÖSUNG MIT BOOLSCHEN GLEICHUNGEN
*X-NAMES
 RD, WR, ADR [0..15]                                                    ;
*Y-NAMES
 RAM, ROM1, ROM2, PIO, SIO                                              ;
*BOOLEAN-EQUATIONS
 RAM     =   ADR15                                                      ;
 /ROM1   =   /RD      & WR      & /ADR13   & ADR14    & ADR15           ;
 /ROM2   =   /RD      & WR      & ADR13    & ADR14    & ADR15           ;
 /PIO    =    RD      & /WR     & /ADR4    & /ADR5    & /ADR6 &
             /ADR7     & /ADRB   & /ADR9    & /ADR10   &
             /ADR11    & /ADR12  & /ADR13   & /ADR14                    ;
 /SIO    =    ADR4    & /ADRS   & /ADR6    & /ADR7    &
             /ADRB     & /ADR9   & /ADR10   & /ADR11   &
             /ADR12    & /ADR13  & /ADR14                               ;
*PLD
 TYPE = HYPERPLD                                                        ;
*RUN-CONTROL
 LISTING = PLOT                                                         ;
*END
```

Als kombinatorisches Beispiel für Boolsche Gleichungen dient ein Dekoder für ein Mikro-

computersystem, der 5 Selektsignale für die Adressierung von RAM, ROM, einer bidirektionalen seriellen (SIO) sowie einer unidirektionalen parallelen Schnittstelle (PIO) liefert (Tab. 10.2). Der Entwurf ist hardwareunabhängig (HYPERPLD) und hat die Darstellung der Schaltung (PLOT) zum Ziel (Fig. 10.50).

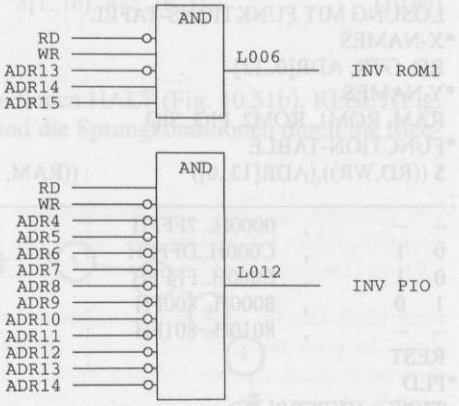


Fig. 10.50
Ausdruck eines Schaltungsausschnittes des mit dem Datensatz von Tab. 10.2 entwickelten Adreßdekoders für ein Mikrocomputersystem (Originalausdruck LOG/iC)

Eingabe über Funktionstabellen (Wahrheitstabellen) Funktionstabellen stellen tabellarisch den Zusammenhang dar zwischen allen Wertekombinationen der Eingangsvariablen (den Eingangsvektoren) und den zugeordneten Werten der Ausgangsvariablen (den Ausgangsvektoren).

Da nicht immer alle Eingangsvariablen für einen Ausgangsvektor relevant sind und gelegentlich auch der Wert einer Ausgangsvariablen beliebig gewählt werden kann, erlaubt die Funktionstafel, diesen Sachverhalt durch sog. Don't Cares (Striche in der Tabelle) zu berücksichtigen. Dabei stehen Don't Cares auf der Eingangsseite für beide Werte 0 und 1 der betreffenden Variablen, gestatten also die Einsparung einer Zeile der Beschreibung. Don't Cares auf der Ausgabeseite sind wichtiger. Sie erhöhen den Freiraum für den Compiler und vereinfachen die Schaltung.

Da Funktionstabellen den Wertebereich des Eingangsvektors immer vollständig erfassen, müssen Werte, über die nicht explizit eine Aussage gemacht wird, in einer letzten Zeile „REST" berücksichtigt werden. Der Entwickler muß festlegen, welchen Wert die Ausgangsvariablen haben sollen, wenn keine der vorangehenden Bedingungen für die Eingangsvariablen erfüllt ist.

Eine vorteilhafte Eigenschaft der Funktionstafel ist, daß Bereiche von Variablen zu Zahlenfeldern zusammengefaßt und dabei alle Zahlensysteme (auch gemischt) Anwendung finden können. Die Kennzeichnung erfolgt durch einen nachgestellten Buchstaben. H steht für Hexadezimal, D für Dezimal, Q für Oktal und B für Binär (B kann aber auch entfallen). Die Anwendung dieser Möglichkeiten wird an der Beschreibung des bereits vorher als Beispiel dienenden Adreßdekoders demonstriert (Tab. 10.3). Entwurfsziel ist diesmal die Gewinnung der Boolschen Gleichungen.

Tab. 10.3 Datensatz für den Adreßdekoder nach Tab. 10.2, jedoch hier mit einer Funktionstafel als Eingabeformat. Ziel der Entwicklung auf logischer Ebene war die Gewinnung der darunter ausgedruckten Boolschen Gleichungen

```
*IDENTIFICATION
 DEKODER FÜR MIKROCOMPUTERSYSTEM
 LÖSUNG MIT FUNKTIONS-TAFEL
*X-NAMES
 RD, GTR, ADR[0..13]                                                    ;
*Y-NAMES
 RAM, ROM1, ROM2, PIO, SIO                                              ;
*FUNCTION-TABLE
 $ ((RD,WR)),(ADR[13..0])          : ((RAM, ROM1, ROM2, PIO, SIO))      ;
;
 –  –      ,  0000H.. 7FFFH   :   0  1  1  –  –   ;  RAM     SEL   ;
 0  1      ,  C000H..DFFFH    :   1  0  1  –  –   ;  ROM1    SEL   ;
 0  1      ,  E000H..FFFFH    :   1  1  0  –  –   ;  ROM2    SEL   ;
 1  0      ,  8000H.. 800FH   :   –  –  –  0  1   ;  PIO     SEL   ;
 –  –      ,  8010H.. 801FH   :   –  –  –  1  0   ;  SIO     SEL   ;
 REST                         :   1  1  1  1  1   ;  NICHTS         ;
*PLD
 TYPE = HYPERPLD
*RUN-CONTROL
 LISTING = EQUATIONS                                                    ;
*END
```

```
        ***********************************************************************
        ***                    BOOLEAN EQUATIONS                            ***
        ***********************************************************************
 RAM    =  ADR15  ;
/ROM1   =  /RD    &  WR     &  /ADR13  &  ADR14   &  ADR15   ;
/ROM2   =  /RD    &  WR     &  ADR13   &  ADR14   &  ADR15   ;
/PIO    =  RD     &  /WR    &  /ADR4   &  /ADR5   &  /ADR6
                  &  /ADR7  &  /ADRB   &  /ADR9   &  /ADR10
                  &  /ADR11 &  /ADR12  &  /ADR13  &  /ADR14   ;
/SIO    =  ADR4   &  /ADR5  &  /ADR6   &  /ADR7   &  /ADRB
                  &  /ADR9  &  /ADR10  &  /ADR11  &  /ADR12
                  &  /ADR13 &  /ADR14                         ;
```

Die FSM-Syntax für Steuerwerke Bei sequentiellen Schaltungen ist es günstig, als ersten Entwicklungsschritt eine grafische Problemdarstellung zu wählen. Die einfachste, i.a. ausreichende Darstellung ist das Blasendiagramm (Bubble Diagramm), das gegenüber anderen Verfahren mit einem einzigen Symbol auskommt. Die Darstellung eines 4-Bit-Zählers ergibt einen geschlossenen Ring mit 16 Zuständen, die nacheinander eingenommen werden. Jeder Zustand hat seine eigene, wählbare Kodierung.

Ein einfacher Aufwärtszähler ohne Sonderfunktionen nach Fig. 10.51 kommt zunächst ohne die rechts stehenden Konditionen für den Zustandswechsel aus und erlaubt die Ableitung der FSM-Syntax.

Mit Si für den aktuellen und Fi+1 für den Folgezustand kann zeilenweise vereinbart werden:

$$\begin{array}{lcl} S_1 & , & F_2 \\ S_2 & , & F_3 \\ \vdots & , & \vdots \\ S_{15} & , & F_{16} \\ S_{16} & , & F_1 \end{array}$$

Der gleiche Sachverhalt läßt sich auch in einer einzigen Zeile zusammenfassen (ein Trivialbeispiel für die FSM-Syntax)

$$S[1..16], F[2..16, 1] \qquad (10.49)$$

Bei einer Erweiterung des Zählers durch die Funktionen HALT (Fig. 10.51b), RESET(Fig. 10.51c), eventuell auch Vor-Rückwärts-Zählen sind die Sprungkonditionen durch die Steuervariablen RES, HOLD und UP festzulegen.

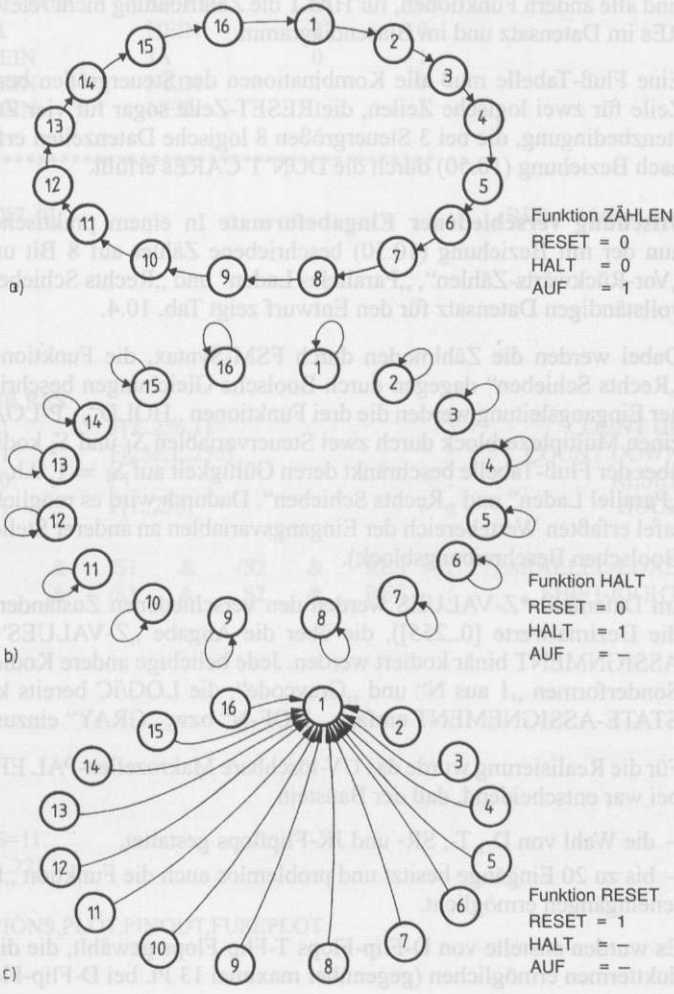


Fig. 10.51
Darstellung der Funktion des 4-Bit-Binärzählers nach Gl. (10.50) durch Blasendiagramme. Drei Steuervariable legen die Sprungkondition beim Zustandswechsel fest

In der FSM-Syntax erfolgt die logische Beschreibung des durch HALT und RESET erweiterten Zählers in Analogie zu den Blasendiagrammen durch eine Flußtafel. In dieser werden die Steuervariablen in einer Spalte „X" in der Reihenfolge RESET, HALT, AUF als Kondition für den spezifischen Folgezustand F_{i+1} eines Zustandes S_i aufgeführt:

```
S[1..16] , X   0   0   1 , F[2..16,1]  ; AUF-ZÄHLEN
S[1..16] , X   0   0   0 , F[16,1..15] ; AB-ZÄHLEN
S[1..16] , X   0   1   - , F[1..16]    ; HALT
S[1..16] , X   1   -   - , F1          ; RESET
```
(10.50)

Die Tatsache, daß nicht alle Steuergrößen von gleicher Wertigkeit sein müssen (für RESET sind alle andern Funktionen, für HALT die Zählrichtung nicht relevant), erlaubt DON'T CAREs im Datensatz und im Blasendiagramm.

Eine Fluß-Tabelle muß alle Kombinationen der Steuergrößen beschreiben. Da die HALT-Zeile für zwei logische Zeilen, die RESET-Zeile sogar für vier Zeilen steht, ist die Konsistenzbedingung, die bei 3 Steuergrößen 8 logische Datenzeilen erfordert, für den Datensatz nach Beziehung (10.50) durch die DON'T CAREs erfüllt.

Mischung verschiedener Eingabeformate In einem praktischen Entwurfsbeispiel soll nun der mit Beziehung (10.50) beschriebene Zähler auf 8 Bit und durch die Funktionen „Vor-Rückwärts-Zählen", „Paralleles Laden" und „Rechts Schieben" erweitert werden. Den vollständigen Datensatz für den Entwurf zeigt Tab. 10.4.

Dabei werden die Zählmoden durch FSM-Syntax, die Funktionen „Parallel Laden" und „Rechts Schieben" dagegen durch Boolsche Gleichungen beschrieben. Zur Einsparung einer Eingangsleitung werden die drei Funktionen „HOLD", „P-LOAD" und „SHIFT-R" über einen Multiplexerblock durch zwei Steuervariablen S_1 und S_2 kodiert. Die Zeile „Relevant" über der Fluß-Tabelle beschränkt deren Gültigkeit auf $S_1 = 1$, d.h. auf alle Funktionen außer „Parallel Laden" und „Rechts Schieben". Dadurch wird es möglich, den nicht von der Flußtafel erfaßten Wertebereich der Eingangsvariablen an anderer Stelle festzulegen (hier in dem Boolschen Beschreibungsblock).

Im Datenfeld *Z-VALUES werden den verschiedenen Zuständen Werte zugewiesen (hier die Dezimalwerte [0..255]), die über die Angabe „Z-VALUES" im Datenfeld *STATE-ASSIGNMENT binär kodiert werden. Jede beliebige andere Kodierung ist möglich. Für die Sonderformen „1 aus N" und „Graycode", die LOG/iC bereits kennt, reicht es aus, unter STATE-ASSIGNEMENT einfach „1-OF-N" bzw. „GRAY" einzusetzen.

Für die Realisierung wurde das UV-löschbare Makrozellen-PAL EP600 (Altera) gewählt. Dabei war entscheidend, daß der Baustein

- die Wahl von D-, T-, SR- und JK-Flipflops gestattet,
- bis zu 20 Eingänge besitzt und problemlos auch die Funktion „Parallel Laden" mit 8 Dateneingängen ermöglicht.

Es wurden anstelle von D-Flip-Flops T-Flip-Flops gewählt, die die Realisierung mit 5 Produkttermen ermöglichen (gegenüber maximal 13 Pt. bei D-Flip-Flops, s. Tab. 10.5).

Tab. 10.4 Datensatz eines 8-Bit-Binärzählers mit Sonderfunktionen. Die Zählfunktionen werden in einer Flußtafel mit der FSM-Syntax beschrieben, die anderen Funktionen im gleichen Datensatz mit Boolschen Gleichungen. Durch die Angabe eines Bausteintyps und seiner Anschlußbelegung erfolgt diesmal die Entwicklung auch auf physikalischer und nicht nur auf logischer Ebene

```
*IDENTIFICATION                                                    ;
 8-Bit-Binärzähler mit Parallel-Load und Shift-Right               ;
 ;
 ************************************************************
 ; Ausführung des Zählers mit dem ALTERA-PAL EP600
 ************************************************************
 ; Zur Einsparung von Anschlüssen werden einige Zählerfunk-
 ; tionen durch Steuerleitungen S1 und S2 gesteuert:
 ;
 ;     HOLD      P-LOAD      SHIFT-R      S1     S2
 ;     JA        JA          NEIN         0      0
 ;     JA        NEIN        JA           0      1
 ;     JA        NEIN        NEIN         1      0
 ;     NEIN      NEIN        NEIN         1      1                ;
 ;
 ************************************************************
*X-NAMES
 S1, 52, UP, RES, D(7..0), Q[7..0]                    ;   RES = Aktiv-Low!
*Z-NAMES
 Q[7..0]                                                           ;
*Z-VALUES
 S[1..256] = [0..255]                                              ;
*FLOW-TABLE
 RELEVANT = S1=1, S2, UP, RES                                      ;
 S[1..256]   ,   X   111   ,   F[2..256,1]                         ;   COUNT UP
 S[1..256]   ,   X   101   ,   F[256,1..255]                       ;   COUNT DOWN
 S[1..256]   ,   X   --0   ,   F1                                  ;   RESET
 S[1..256]   ,   X   0-1   ,   F[1..256]                           ;   HOLD
*BOOLEAN-EQUATIONS
 Q[7..0] := D[7..0]          &   /S1   &   /S2   &   RES   ;   PARALLEL-LOAD
 Q[7..0] := Q[6..0,7]        &   /S1   &    S2   &   RES   ;   SHIFT-RIGHT
*STATE-ASSIGNMENT
 Z-VALUES                                                          ;
*FLIPFLOPS
 T-FLIPFLOPS                                                       ;
*PLD
 TYPE = EP600                                                      ;
*PINS
 D[0..7]=[2..9], UP=10, RES=11,
 S1=14, 52=23, Q[7..0]=[15..22]                                    ;
*RUN-CONTROL
 LISTING       = EQUATIONS,PLOT,PINOUT,FUSEPLOT                    ;
 PROGFORMAT = JEDEC                                                ;
*END
```

Tab. 10.5 Die Tabellen zeigen als Ergebnis einer Entwicklung des Zählers nach Tab. 10.4 auf logischer Ebene (mit einem Baustein HYPER-PAL) den Vorteil von T-FlipFlops (rechts) gegenüber D-Flip-Flops (links) im Hinblick auf die erforderliche Zahl von Produktterm en

FUNCTION	INV	P-TERMS	FUNCTION	INV	P-TERMS
Q7.D	NO	13 *	Q7.T	NO	5
	YES	14 *		YES	13 *
Q6.D	NO	12 *	Q6.T	NO	5
	YES	13 *		YES	12 *
Q5.D	NO	11 *	Q5.T	NO	5
	YES	12 *		YES	11 *
Q4.D	NO	10 *	Q4.T	NO	5
	YES	11 *		YES	10 *
Q3.D	NO	9	Q3.T	NO	5
	YES	10		YES	9
Q2.D	NO	8	Q2.T	NO	5
	YES	9		YES	8
Q1.D	NO	7	Q1.T	NO	5
	YES	8		YES	7
Q0.D	NO	4	Q0.T	NO	4
	YES	5		YES	5

Grafische Eingabeformate Als Option lieferbar sind auch zwei grafische Eingabeformate:

1. Eine grafische Stromlaufeingabe ist möglich über Schaltbilder, die mit einem Standardzeichenprogramm wie ORCAD oder PCAD erzeugt wurden. Ein Übersetzungsprogramm erzeugt daraus ein Texteingabeformat, das normal weiterverarbeitet werden kann.
Der Nutzen dieses Eingabeformats darf nicht überbewertet werden. Bereits die Umsetzung vorhandener Schaltungen scheitert häufig daran, daß diese nicht in synchroner Form vorliegen. Noch weniger lohnend ist es, einen neuen Entwurf auf dem Umweg über ein Schaltbild durchzuführen.

2. Das zweite grafische Eingabeformat heißt STATE/view. Die Eingabe erfolgt über ein Ablaufprogramm, das mit einem grafischen Editor aus einer Bibliothek genormter Elemente erzeugt wird. Das fertige Ablaufdiagramm wird anschliessend in ein Texteingabeformat übersetzt.
Der Nutzen von STATE/view für die Formulierung komplexer Steuerwerke (Finite State Machines) wird hoch eingestuft, weil es die Entwicklungsarbeit um einen Schritt reduziert.

Der Entwurf komplexer Schaltungen beginnt in der Regel mit einer Skizze des Ablaufschemas, dessen mühsame Umsetzung dem Entwickler durch STATE/view erspart wird.

10.3.3.3 Ausgabeformate und Dokumentation

LOG/iC bietet als Ergebnis der Entwicklung eine Fülle unterschiedlicher Ausgabedateien an. Dabei sind nur drei auf einen bestimmten Baustein bezogen und ausschließlich über einen Datensatz zu gewinnen, der Angaben über den konkreten Bausteintyp und die Belegung seiner Anschlüsse enthält. Es sind dies

– das Programmier-File für PALs und PLAs (JEDEC-Format, s. Tab. 10.6a) bzw. für PROMs (INTEL-HEX-Format) zur Steuerung des angeschlossenen Programmiergeräts. Es handelt sich dabei um zwei Formate, die zwar keine echte Norm darstellen, sich aber als sogenanntes Industrie-Standard weltweit gegen andere Formate durchgesetzt haben.

– die sogenannte `Fusematrix` (FUSE-PLOT), eine Liste der intakten und geöffneten Sicherungen mit ihren Nummern (s. Tab. 10.6b).

– das Anschlußbild des Bausteins (PINPLOT) mit eingetragenen Namen der Signale an den Anschlüssen. Falls ein Baustein in mehreren Gehäuseformen geliefert ist, werden alle Alternativen dargestellt (s. Fig. 10.52).

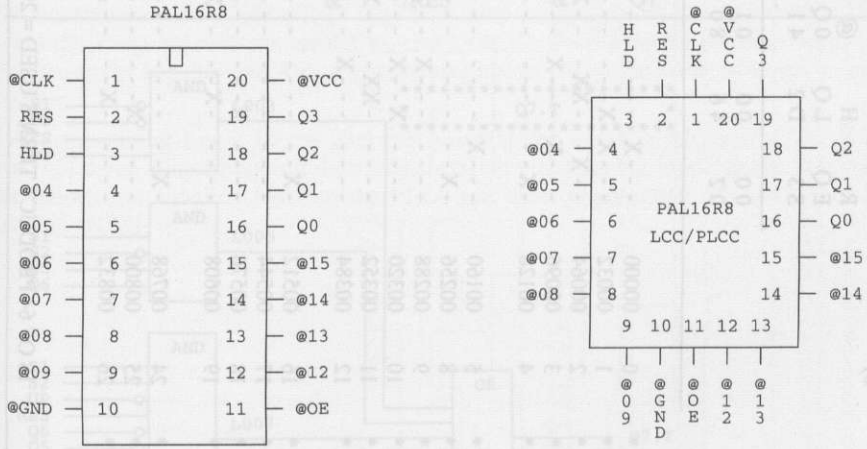


Fig. 10.52 Auch die Anschlußbelegung des Bausteins in allen lieferbaren Ausführungen ist das Ergebnis der Entwicklung (Originalausdruck LOG/iC)

Ob diese Dateien erzeugt werden oder nicht, bestimmt der Entwickler im Datenfeld *RUN-CONTROL.

Bereits als Ergebnis einer Entwicklung auf logischer Ebene für einen hypothetischen und restriktionsfreien Baustein HYPERPLD sind die folgenden Ausgabedateien zu gewinnen:

– Die Lösungsbeschreibung mit Boolschen Gleichungen in Normalform. LOG/iC ist immer (unabhängig vom Eingabeformat und bei beliebiger Komplexität) in der Lage, beliebige Probleme mit Gleichungen darzustellen (Tab. 10.7). Das ermöglicht dem Entwickler, interaktiv zu entwickeln und z.B. mit Test-Eingabedateien Bildungsgesetze von Zählern mit ungewöhnlicher Kodierung abzuleiten.

– Die Lösungsbeschreibung mit Schaltbildern (Fig. 10.53). Viele Praktiker ziehen die grafische Darstellung jeder anderen Darstellungsform vor.

Tab. 10.6 Das JEDEC-Programmierfile (a) entspricht weitgehend der Sicherungsmatrix des Bausteins (b) nach seiner Programmierung.

a)

```
D2224*
GO*
QP020*
QF2O4B*
FO*
L0000 11101111111111101111111111111111*
L0032 11100111111111111111111111111111*
L0064 11011001110111011111111111111111*
L0096 11101110111111111111111111111111*
L0128 10111111111111111111111111111111*
L0160 11101111111011111111111111111111*
L0256 10111111111111111111111111111111*
L0288 11111110111111101111111111111111*
L0320 11110110111111111111111111111111*
L0352 11111001110111011111111111111111*
L0384 11111110111011111111111111111111*
L0512 10111111111111111111111111111111*
L0544 11111111111011101111111111111111*
L0576 11110111111011111111111111111111*
L0608 11111011110111011111111111111111*
L0768 10111111111111111111111111111111*
L0800 11110111111111101111111111111111*
L0832 11111011111111011111111111111111*
C4120*
000000
```

b)

			R EQ S3	H LQ D2	@ 0Q 41	@ 0Q 50	@@ 01 65	@@ 01 74	@@ 01 83	@@ 01 92	
			00 02	00 46	01 80	11 24	11 68	22 02	22 46	23 80	
L0000	0	00000	---X	----	----	---X	----	----	----	----	Q3
L0032	1	00032	---X	X---	----	----	----	----	----	----	
L0064	2	00064	--X-	-XX-	--X-	--X-	----	----	----	----	
L0096	3	00096	---X	---X	----	----	----	----	----	----	
L0128	4	00128	-X--	----	----	----	----	----	----	----	
L0160	5	00160	---X	----	---X	----	----	----	----	----	Q2
L0256	8	00256	-X--	----	----	----	----	----	----	----	
L0288	9	00288	----	---X	----	---X	----	----	----	----	
L0320	10	00320	----	X--X	----	----	----	----	----	----	
L0352	11	00352	----	-XX-	--X-	--X-	----	----	----	----	
L0384	12	00384	----	---X	---X	----	----	----	----	----	
L0512	16	00512	-X--	----	----	----	----	----	----	----	Q1
L0544	17	00544	----	----	---X	---X	----	----	----	----	
L0576	18	00576	----	X---	---X	----	----	----	----	----	
L0608	19	00608	----	-X--	--X-	--X-	----	----	----	----	
L0768	24	00768	-X--	----	----	----	----	----	----	----	Q0
L0800	25	00800	----	X---	----	---X	----	----	----	----	
L0832	26	00832	----	-X--	----	--X-	----	----	----	----	

18 OF 64 PRODUCT TERMS USED = 29 %

Tab. 10.7 Die Boolsche Gleichung der letzten Stufe des Zählers nach Tab. 10.4 als Ausgabe der Entwicklung. Eingabeformat war eine Flußtafel in FSM-Syntax. LOG/iC kann beliebig komplexe Probleme unabhängig vom Eingabeformat durch Boolsche Gleichungen beschreiben

```
************************************************************
***                  BOOLEAN EQUATIONS                   ***
************************************************************
;
; ATTENTION !!! T-FLIPFLOP REQUIRED FOR Q7.T
;
Q7.T    :=   /S1   &   /S2   &   RES   &   D7
        +    /S1   &   S2    &   RES   &   Q6
        +    S1    &   S2    &   /UP   &   RES
                   &   /Q6   &   /Q5   &   /Q4
        +    S2    &   /Q3   &   /Q2   &   /Q1
                   &   /QO
                   &   UP    &   RES   &   Q6
                   &   Q5    &   Q4    &   Q3
        +    S1    &   Q2    &   Q1    &   QO
                   &   /RES  &   Q7    ;
```

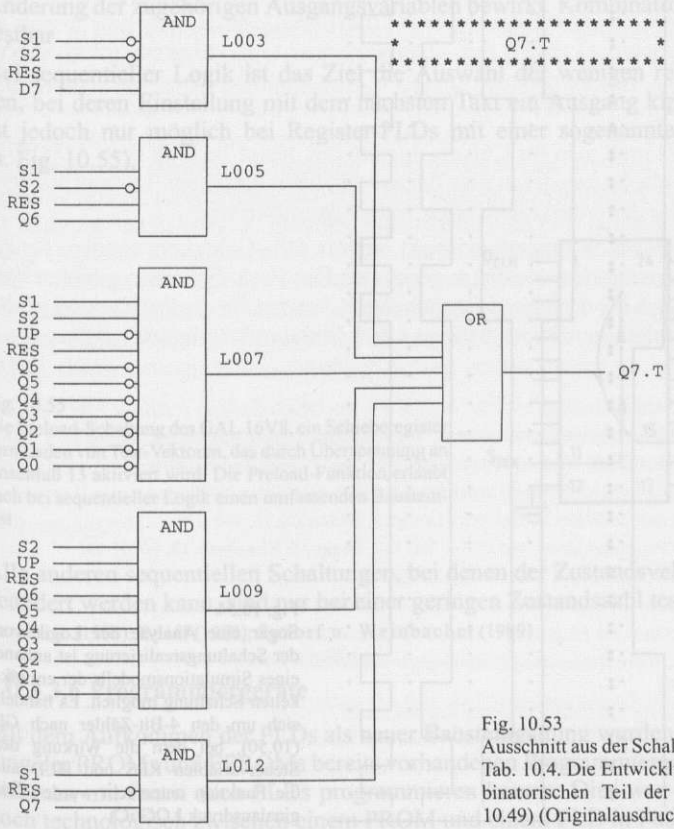


Fig. 10.53
Ausschnitt aus der Schaltung des 8-Bit-Binärzählers nach Tab. 10.4. Die Entwicklungssoftware stellt nur den kombinatorischen Teil der FSM-Schaltung dar (vgl. Fig. 10.49) (Originalausdruck LOG/iC)

10.3.3.4 Logik-Analyse an einem Simulationsmodell

Es ist äußerst hilfreich, wenn der Anwender die Funktion einer Schaltung bereits zu einem sehr frühen Zeitpunkt anschaulich darstellen und beurteilen kann. Das gilt besonders für Konzepte, die erst während ihrer Entwicklung zur endgültigen Form gelangen (interaktiver Entwurf). LOG/iC unterstützt ein solches Vorgehen des Entwicklers sehr weitgehend. Mit dem als Option lieferbaren Functional Verifier ist es sogar möglich, bereits in der Phase der logischen Entwicklung anhand eines Simulationsmodells eine Logik-Analyse durchzuführen. Dabei ist es (wie bei der Logik-Analyse einer tatsächlich vorhandenen Schaltung) möglich, Steuergrößen einzustellen und die Auswirkung auf die Schaltungsfunktion zu überprüfen (Fig. 10.54).

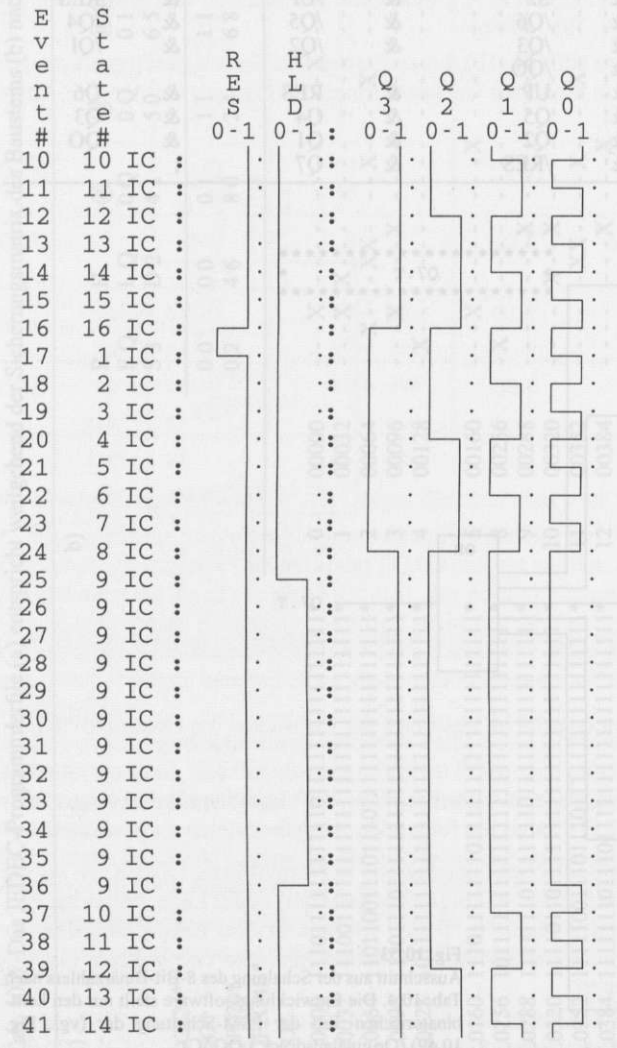


Fig. 10.54
Sogar eine Analyse der Logik von der Schaltungsrealisierung ist anhand eines Simulationsmodells der entwickelten Schaltung möglich. Es handelt sich um den 4-Bit-Zähler nach Gl. (10.50), bei dem die Wirkung der Steuervariablen RES und HLD auf die Funktion untersucht wurde (Originalausdruck LOG/iC)

10.3.3.5 Bausteintest mit Testvektoren

PLD-Schaltungen werden in der Regel funktional getestet, d.h., eingesetzt in die Gesamtschaltung. Dieses Verfahren versagt, wenn die Gesamtschaltung erst später oder nur an einem anderen Ort zur Verfügung steht.

Prinzipiell ist eine umfassende Beurteilung eines kombinatorischen PLD-Bausteins auch dadurch möglich, daß in einer Testschaltung für alle denkbaren Bitmuster der Eingangsvariablen die Reaktion der Bausteinausgänge festgestellt wird. Dies ist jedoch bei 2^n Möglichkeiten für n Eingangsvariable ein sehr unpraktikables Verfahren. Ganz unmöglich ist mit diesem Verfahren der Test sequentieller Schaltungen. Es müssen ja nicht nur die Eingangsvariablen, sondern für jede ihrer Kombinationen zusätzlich die Zustandsvektoren des Netzwerks variiert werden.

Der Test mit Testvektoren basiert auf der Überlegung, daß nur ein geringer Teil der möglichen Änderungen des Eingangs- und des Zustandsvektors tatsächlich Reaktionen an den Bausteinausgängen hervorruft. Diese Vektoren werden automatisch erzeugt und an das JEDEC-Programmierfile angehängt. Der Test erfolgt im Programmiergerät unmittelbar nach Abschluß der Programmierung.

Für kombinatorische Logik erfolgt die Testvektorerzeugung produkttermorientiert. Die Auswahl der Eingangsvektoren erfolgt so, daß einer der Produktterme aktiviert wird und eine Änderung der zugehörigen Ausgangsvariablen bewirkt. Kombinatorische Logik ist zu 100 % testbar.

Bei sequentieller Logik ist das Ziel die Auswahl der wenigen relevanten Zustandsvektoren, bei deren Einstellung mit dem nächsten Takt ein Ausgang kippt. Diese Voreinstellung ist jedoch nur möglich bei Register-PLDs mit einer sogenannten PRELOAD-Schaltung (s. Fig. 10.55).

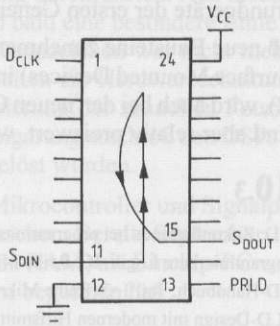


Fig. 10.55
Die Preload-Schaltung des GAL 16V8, ein Schieberegister zum Laden von Test-Vektoren, das durch Überspannung an Anschluß 13 aktiviert wird. Die Preload-Funktion erlaubt auch bei sequentieller Logik einen umfassenden Bausteintest

Alle anderen sequentiellen Schaltungen, bei denen der Zustandsvektor nur durch Einzählen geändert werden kann, sind nur bei einer geringen Zustandszahl testbar.
Tatje u.a. (1986); Reiff (1989); Sadorf u. Weinbacher (1989)

10.3.3.6 Programmiergeräte

Mit dem Aufkommen der PLDs als neuer Bausteingattung wurden die für die Programmierung der PROMs und EPROMs bereits vorhandenen Programmiergeräte so weiterentwickelt, daß man mit ihnen auch PLDs programmieren konnte. Dies war einfach möglich, besteht doch technologisch zwischen einem PROM und einem PLD in Fuse-Link-Technik kein Un-

terschied. Diese Geräte bestanden aus einem Grundgerät und einem Satz bausteinspezifischer Prüfadapter (sog. Personality Moduls) mit einer für die jeweilige Bausteinfamilie speziellen Anpassungselektronik.

Das Grundgerät enthielt neben der Stromversorgung alle zur Bedienung notwendigen Komponenten, mindestens also eine Eingabetastatur und eine alphanumerische Anzeige. Eine Rechnerschnittstelle war, wenn überhaupt vorhanden, eher nebensächlich.

Es muß dringend davon abgeraten werden, solche Universal-Programmierer, die der Markt immer noch anbietet, zu kaufen. Da die Zahl unterschiedlicher Bauteile rasch wächst, steigt auch die Zahl der Prüfadapter, die der Anwender für seine Arbeit benötigt, schnell an. Das führt zu immensen Folgekosten beim Einsatz dieser Geräte. Dessenungeachtet ist man mit ihnen doch nie auf dem aktuellem Stand der PLD-Technik.

Seit einigen Jahren gibt es eine neue Art von Programmiergeräten, die flexibler der Weiterentwicklung der Bauteile folgen kann. Bei diesen neuen Geräten ist die Bedienung nur über den angeschlossenen Rechner möglich. Das führt nicht nur zur Einsparung der Kosten für die Tastatur und Anzeige und zur Steigerung des Bedienungskomforts, sondern vor allem zur problemlosen Anpassung der Geräte an unterschiedliche Programmieralgorithmen über die Rechnersoftware. Der wesentlichste Fortschritt dieser Geräte besteht aber im Konzept der sogenannten Pintreiber. Man versteht darunter eine Anpassungselektronik, die für jeden Anschluß der Prüffassung mit Ausnahme der Stromversorgungsanschlüsse einen vollständigen Impulsgenerator und einen Analog-Digital-Wandler enthält. Diese Pintreiber sind in der Lage, Spannungen zwischen 0 V und +25 V (bei einigen Geräten sogar −36 V bis +40 V !) in 0,1 V-Schritten nicht nur zu liefern, sondern für die Verifikation des Bausteins auch digital messen zu können.

Nach dem Stand von 1992 muß man für Universalgeräte mit dieser Technik etwa 4 000 DM aufwenden. Die neuen Geräte kosten damit trotz der aufwendigen Pintreiber-Technik nur etwa soviel wie die Grundgeräte der ersten Generation von Programmierern.

Das Problem, daß neue Bausteine zunehmend nicht mehr in DIL-Gehäusen, sondern für die SMD-Technik (Surface Mounted Devices) in sogenannten Flat-Pack-Gehäusen geliefert werden (s. Fig. 10.52), wird auch bei den neuen Geräten durch Adapter für die Prüffassung gelöst. Diese Adapter sind aber relativ preiswert, weil sie keine Elektronik enthalten müssen.

Literatur zu 10.3

Agrawal, O.P. (1991): Zukunftstrends bei programmierbarer Logik. Elektronik Informationen **7**, 37–39

Auer, A. (1990): Programmierbare Logik–IC. Reihe Mikroelektronik, Bd. 3. Heidelberg: Hüthig

Auer, A. (1991): PLD-Handbuch, Teil 1–3. Reihe Mikroelektronik, Bd. 6. Heidelberg: Hüthig

Göritz, J. (1989): PLD-Design mit modernen Hilfsmitteln. Design & Elektronic **11**, 146–148

Mehlo, H. (1989): Aufbau und Programmierung von PALs, Teil 1–2. Design & Elektronik **18**, 60–62 usf.

Petersen, K. (1988): Programmierbare Logik. Elektronik industrie **11**, 77–78 usf.

Reiff, E.C. (1988): PLD-Compiler mit Logiksynthese. Design & Elektronik **24**, 142–144

Reis, W. (1989): Zustandsmaschinen aus programmierbaren Logikbausteinen. Elektronik Informationen **9**, 70–72

Sadorf, H.; Weinbacher, T. (1989): Realisierungsunabhängiger Elektronik-Entwurf. Design & Elektronik **24**, 156–161

Schimmer, F. (1990): Optimierung von PLD. Elektronik Informationen 1, 64–65

Tatje, J. (1989): Optimierung – immer noch unverzichtbar. Elektronik industrie **5**, 62–64

Tatje, J.; Biel, G. (1986): Neue Perspektiven für PLDs durch den universellen Logik-Compiler LOG/iC. Der elektroniker **6**, 50–62

Tatje, J.; Ditzinger, A. (1986): PLD-Design auf dem PC. Design & Elektronik **9**, 128–132