Kapitel 4
Programmierbare Logik

Ausgabestand 1.1

- Nur zur Information -

4. Programmierbare Logik

4.1. Grundlagen

4.1.1. Weshalb programmierbare Logik?

Die Alternativen: wir können ein System (1) auf Grundlage fertiger (Off the Shelf) Schaltkreise aufbauen (diese enthalten elementare Digitalschaltungen, wie Gatter, Flipflops, Zähler usw.), oder wir können (2) das System zunächst einmal als Zusammenschaltung von Gattern, Flipflops usw. entwerfen und diesen Entwurf in kundenspezifischen Schaltkreisen (ASICs) verwirklichen lassen. Die Vor- und Nachteile beider Ansätze sind offensichtlich (Tabelle 4.1).

Prinzip	Off-the-Shelf-Schaltkreise	ASICs
Realisierung der gewünschten Funktionen	durch Leiterplattenfertigung; notfalls auch durch Verdrahtung von Hand ("am Küchentisch")	erfordert Halbleiterfertigung
Änderbarkeit	auf der Leiterplatte - unproblematisch und schnell (einfaches Handwerk)	in jeder Hinsicht aufwendig (neuer Schaltkreis)
Gebrauchseigenschaften	eher mäßig im Vergleich zu ASIC-Lösungen (größer, höherer Stromverbrauch, langsamer, geringere Zuverlässigkeit)	hervorragend (State of the Art)
Eignung	für eher geringe Stückzahlen bis hin zur Einzelfertigung	für eher hohe (typischerweise wenigstens fünfstellige) Stückzahlen

Tabelle 4.1 Off-the-Shelf-Schaltkreise und ASICs: ein Vergleich

Mit programmierbarer Logik wollen die Schaltkreishersteller gleichsam einen Mittelweg eröffnen: Verfügbarkeit "aus dem Regal", einfache Entwicklungsabläufe, leichte Änderbarkeit, Wirtschaftlichkeit selbst bei Einzelfertigung vereinigt mit den Vorzügen von ASIC-Lösungen. Hierfür ist zweierlei notwendig:

- die Festlegung der endgültigen Funktion des Schaltkreises darf keine Halbleiter-Fertigungsschritte mehr erfordern,
- der Schaltkreis muß sich kostengünstig fertigen lassen. Das betrifft sowohl die Ausnutzung der Siliziumfläche als auch die Gehäusekosten. (Der programmierbare Schaltkreis darf nicht teurer sein als die herkömmlichen Schaltkreise, die er ersetzen soll.)

4.1.2. Die schaltalgebraischen Grundlagen

Universelle kombinatorische Schaltungen

Die entscheidende Grundlage bildet die Darstellung der zu realisierenden Schaltfunktion als Wahrheitstabelle oder als disjunktive Normalform DNF (Sum-of-Products-Darstellung; SOP). Wir wissen, daß kanonische disjunktive Normalform (KDNF) und Wahrheitstabelle unmittelbar einander entsprechen. Jede Zeile der Wahrheitstabelle, in der die Schaltfunktion eine Eins als Ergebnis liefert, steht für einen Konjunktions- bzw. Produktterm. Alle Produktterme werden disjunktiv verknüpft. Dies ist die Grundlage für zwei Ansätze, um universelle kombinatorische Schaltungen zu bauen:

1. wir speichern die gesamte Wahrheitstabelle und sehen für jede gegebene Belegung nach, welches Ergebnis sie liefert,
2. wir bauen eine universelle KDNF-Hardware. Wenn wir jede beliebige Verknüpfung zwischen n Variablen verwirklichen wollen, müssen wir mit allen Möglichkeiten rechnen. Es gibt maximal 2^n Konjunktionsterme (technisch: UND-Gatter mit jeweils n Eingängen), die über eine Disjunktion (technisch: ein ODER-Gatter) mit maximal 2^n Eingängen miteinander verknüpft sind. Zudem muß jede Eingangsvariable sowohl direkt als auch negiert bereitgestellt werden. Nehmen wir an, wir hätten ein solches Netzwerk aufgebaut (Abbildung 4.1). Wäre das ODER-Gatter mit allen UND-Gattern fest verbunden, so hätte das Netzwerk einen ziemlich bescheidenen Funktionsumfang: es würde auf alle 2^n möglichen Eingangsbelegungen hin eine feste Eins liefern. Um eine bestimmte - brauchbare - Funktion zu verwirklichen, müssen wir die Verbindungen zwischen jenen UND-Gattern, die den jeweiligen Produkttermen entsprechen, und dem ODER-Gatter bestehen lassen und alle anderen einfach abschneiden (unter der Annahme, daß ein offener Eingang am ODER-Gatter den logischen Wert "0" repräsentiert).

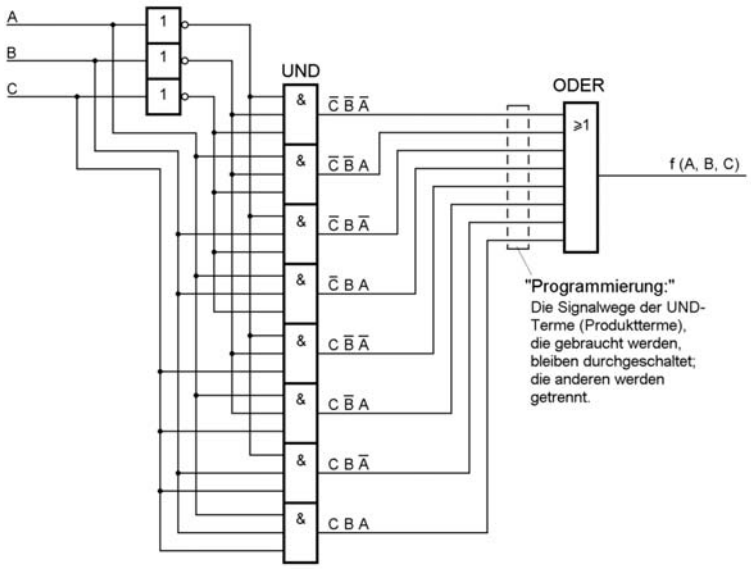


Abbildung 4.1 Eine universelle Kombinatorik-Hardware auf Grundlage der KDNF

Wir wollen uns zunächst danach umsehen, ob es sinngemäß nutzbare Schaltungen schon gibt. Dann wollen wir überlegen, wie wir den offensichtlich hohen Aufwand (2^n UND- Gatter usw.) vermindern können.

Der Decoder als Grundlage einer universellen kombinatorischen Schaltung

Es gibt eine Grundschaltung, in der - bei n Eingängen - 2^n UND-Gatter so verschaltet sind, wie dies in Abbildung 4.1 gezeigt ist: den 1-aus-n-Decoder. Seine Ausgangssignale kennzeichnen alle 2^n überhaupt vorkommenden Eingangsbelegungen. Wir müssen also einem solchen Decoder nur ein ODER-Gatter nachschalten, um eine beliebige Schaltfunktion über die n Eingangsvariablen zu verwirklichen (Abbildung 4.2).

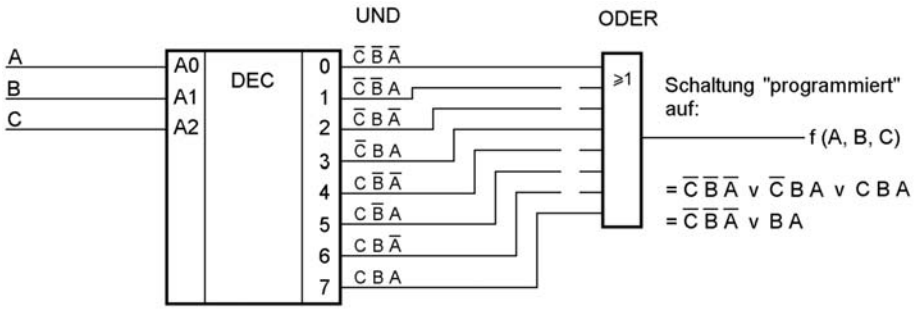


Abbildung 4.2 Der Decoder als Grundlage einer universellen kombinatorischen Schaltung

Nun wollen wir uns zwei Schaltungen ansehen, die Decoder enthalten.

1. Der Multiplexer

Der Multiplexer ist eine Kombination aus einem Auswahlnetzwerk und einem Decoder (der Auswahladresse). Ein Multiplexer mit n Adreßeingängen erlaubt es, einen von 2^n Daten-Eingängen auf den Ausgang durchzuschalten. Wir können somit einen 2^n-zu-1-Multiplexer verwenden, um jede beliebige Schaltfunktion über n Variable zu verwirklichen. Wir müssen dazu nur die Dateneingänge mit den Festwerten "0" oder "1" beschalten, je nach den Ergebniswerten in der Wahrheitstabelle (Abbildung 4.3).

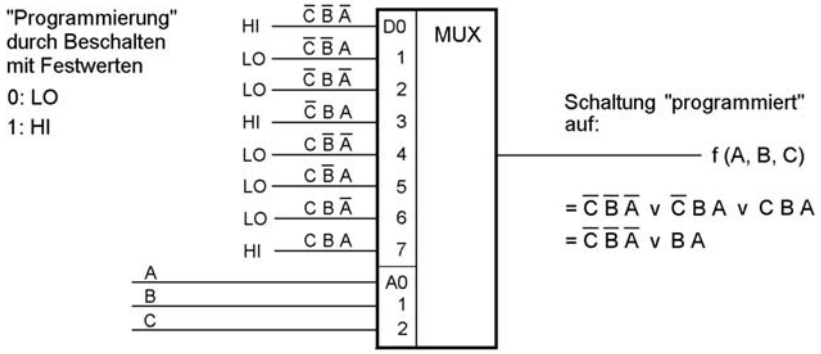


Abbildung 4.3 Verwirklichung einer Schaltfunktion mittels Multiplexer (1)

Man kann sogar die Variablen-Anzahl noch um 1 erhöhen: ein Multiplexer mit n Adreßeingängen kann beliebige Schaltfunktionen mit n + 1 Variablen verwirklichen. Mit anderen Worten: ein n-zu-1-Multiplexer ermöglicht es, eine beliebige Schaltfunktion mit (ld n) +1 Variablen zu realisieren (also beispielsweise ein 8-zu-1-Multiplexer eine Funktion mit 4 Variablen).

Voraussetzung dafür ist, daß jeder Dateneingang gemäß der Wahrheitstabelle mit jeweils einem von vier Werten belegt wird (Abbildung 4.4):

- mit dem Festwert 0,
- mit dem Festwert 1,
- mit einer der Eingangsvariablen,
- mit der besagten Eingangsvariablen in negierter Form.

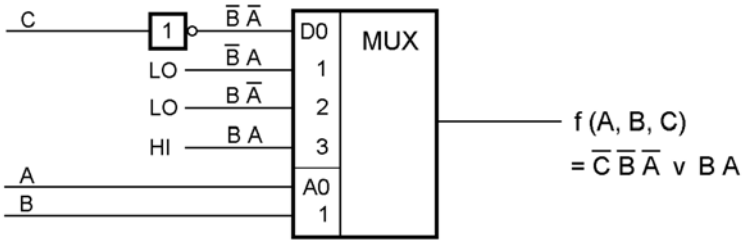


Abbildung 4.4 Verwirklichung einer Schaltfunktion mittels Multiplexer (2)

2. Der adressierbare Speicher
Jede Speicherzelle wird durch Adressierung ausgewählt. n Adreßbits entsprechen 2^n Speicherzellen. Dafür braucht man einen *Adreßdecoder*. Das ist gleichsam die UND-Gatter-Ebene unserer KDNF-Hardware von Abbildung 4.1. Dem ODER-Gatter entspricht der Lesesignalweg, der durch alle Speicherzellen geführt ist. Indem wir die Eingangsvariablen als Adresse verwenden und alle Speicherzellen entsprechend der Wahrheitstabelle mit Nullen oder Einsen füllen, läßt sich somit jede beliebige Schaltfunktion im Rahmen der gegebenen Speicherkapazität verwirklichen (Abbildung 4.5). In anderer Sichtweise: ein adressierbarer Speicher kann die gesamte Wahrheitstabelle aufnehmen.

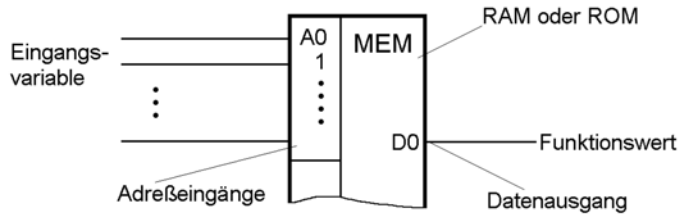


Abbildung 4.5 Verwirklichung einer Schaltfunktion mittels eines adressierbaren Speichers

Ob wir einen ladbaren Speicher (RAM) oder einen Festwertspeicher (ROM) verwenden, ist an sich gleichgültig. Beim RAM haben wir lediglich das technische Problem, wie wir nach jedem Einschalten die Nullen und Einsen immer wieder hineinbekommen (Anfangsladen).

Aufwandsersparnis

2^n UND-Gatter bedeuten, namentlich wenn n größer wird, einen beachtlichen Aufwand. Was kann man tun, um den Aufwand zu verringern? - Viele Schaltfunktionen sind nur für einen sehr geringen Teil der 2^n möglichen Belegungen erfüllt. Man braucht also eigentlich nur entsprechend wenige UND-Gatter. Die Eingangszahl des ODER-Gatters verringert sich dann sinngemäß. Notwendig ist aber, zu bestimmen, ob die Eingänge der UND-Gatter mit den Eingangssignalen direkt oder negiert oder gar nicht*) beschaltet werden (Abbildung 4.6).

*): falls die Variable in dem betreffenden Produktterm gar nicht vorkommt (Don't Care).

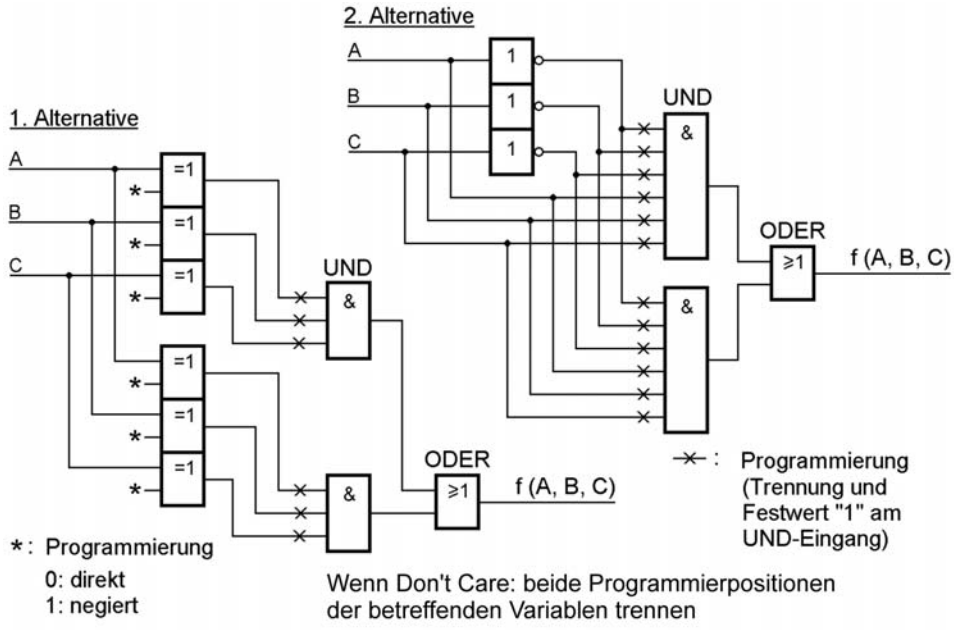


Abbildung 4.6 Aufwandsverminderung der DNF-Hardware

Die Abbildung veranschaulicht 2 Alternativen: (1) die Signale werden mittels programmierbarer Antivalenzgatter invertiert (Komplementbildung), (2) die Signale sind sowohl direkt als auch invertiert durch Programmierung an die UND-Gatter anschaltbar. Beide Lösungen werden in programmierbaren Schaltkreisen verwendet (Lösung (2) häufiger).

Mehrere Schaltfunktionen

Nahezu immer braucht man mehr als eine Schaltfunktion, und oft sind mehrere Schaltfunktionen über dieselben Eingangsvariablen zu bilden. Es sind also mehrere ODER-Gatter mit unabhängigen Ausgängen erforderlich. Im Fall des adressierbaren Speichers entspricht dies einer Vergrößerung der Zugriffsbreite; für "echte" UND-ODER-Strukturen (Schaltungen nach Abbildung 4.6) gibt es zwei Auslegungsmöglichkeiten (Abbildung 4.7):

1. die ODER-Gatter sind fest mit jeweils einigen der UND-Gatter verbunden,
2. die Verbindungen der ODER-Gatter mit den UND-Gattern sind programmierbar.

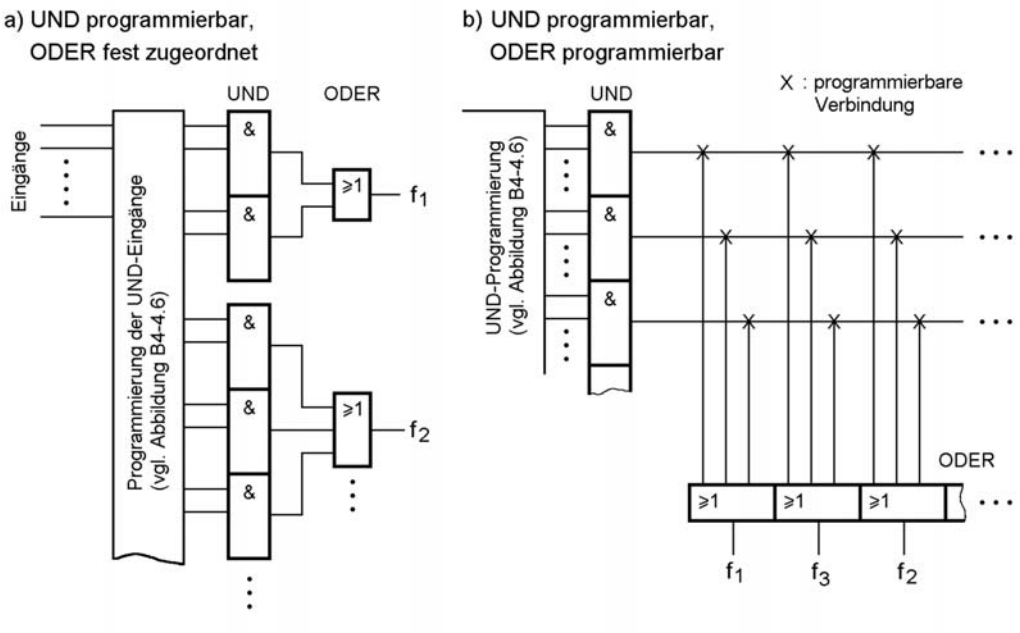


Abbildung 4.7 Realisierung mehrerer Schaltfunktionen

Universelle Funktionsblöcke

Es liegt nahe, komplexe Schaltungen aus mehreren - jeweils einfacheren - Teilschaltungen zusammenzusetzen. Der nächste Schritt: diese Teilschaltungen universell auszugestalten. Das heißt, es geht um Funktionsblöcke mit vergleichsweise wenigen Ein- und Ausgängen, die durch passende Beschaltung bzw. durch Programmierung genutzt werden können, eine Vielzahl von Funktionen zu realisieren. Abbildung 4.8 zeigt ein "klassisches" Beispiel.

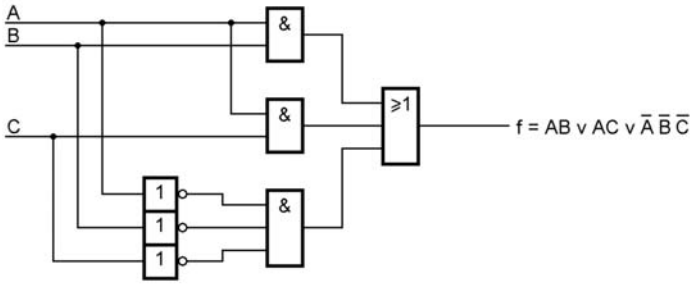
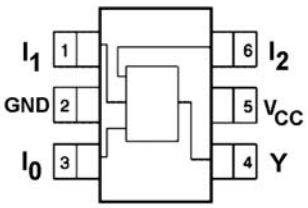


Abbildung 4.8 Universeller kombinatorischer Funktionsblock

Die drei Eingänge dieses Blockes können beliebig mit Festwerten "0" und "1" oder mit irgendwelchen Variablen beschaltet werden.

Dieser Block kann 62 der insgesamt 256 (= 2^{2^3}) möglichen Funktionen von 3 Variablen verwirklichen, darunter alle 16 Funktionen von 2 Variablen. Hierbei wird allerdings vorausgesetzt, daß die einzelnen Variablen je nach Bedarf direkt oder negiert zugeführt werden. (Der Beweis dieser Aussage, der einschließt, alle 62 Funktionen tatsächlich anzugeben, gehört zur "höheren Mathematik" der Schaltalgebra.) Eine akademische Übung? - Keineswegs (Abbildung 4.9).

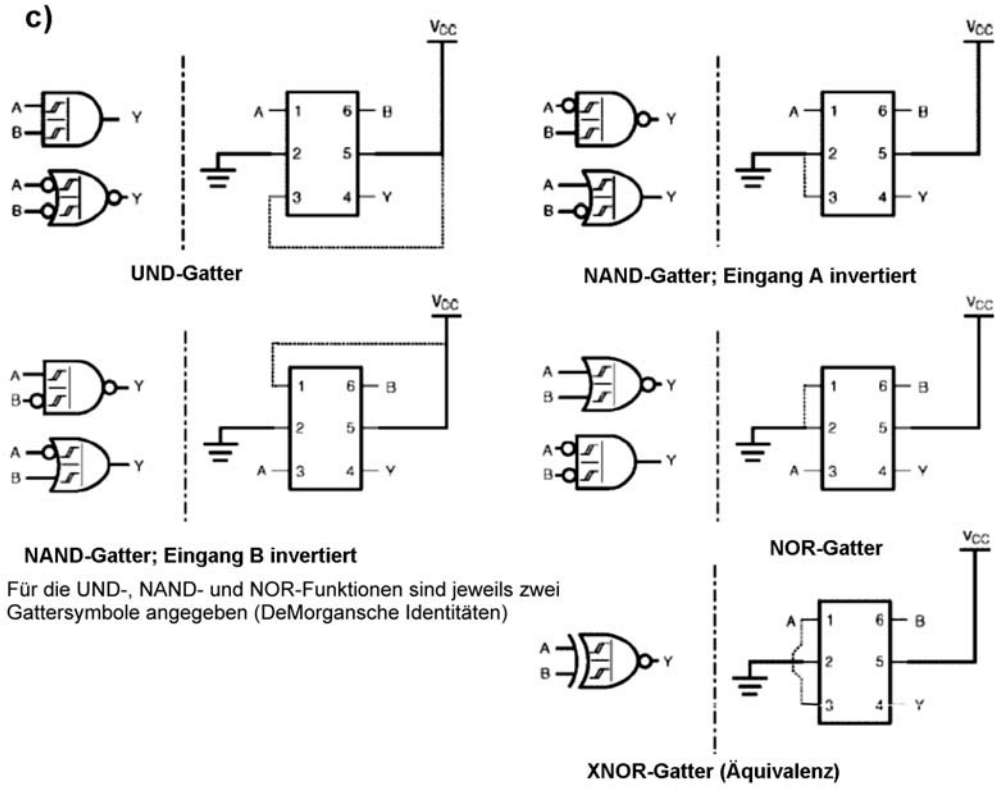
a)



b)

Eingänge			NC7SZ57	NC7SZ58
I_2	I_1	I_0	$Y = (\bar{I}_0)\bullet(\bar{I}_2)+(I_1)\bullet(I_2)$	$Y = (I_0)\bullet(\bar{I}_2)+(\bar{I}_1)\bullet(I_2)$
L	L	L	H	L
L	L	H	L	H
L	H	L	H	L
L	H	H	L	H
H	L	L	L	H
H	L	H	L	H
H	H	L	H	L
H	H	H	H	L

c)



Für die UND-, NAND- und NOR-Funktionen sind jeweils zwei Gattersymbole angegeben (DeMorgansche Identitäten)

Abbildung 4.9 Universelle Funktionsblöcke als Schaltkreise (Fairchild Semiconductor)

Erklärung:

a) Anschlußschema. Es handelt sich darum, einen vielseitigen kleinen Schaltkreis anzubieten, der als Restlogik überall dort eingesetzt werden kann, wo eine einfache kombinatorische Verknüpfung benötigt wird. Der Schaltkreis hat 3 Eingänge I_2, I_1, I_0 und einen Ausgang Y. Wie beim Funktionsblock von Abbildung 4.8 können die Eingänge mit Signalen oder mit Festwerten beschaltet werden.

b) die Funktionstabelle für beide Schaltkreise (nicht darüber nachdenken, wie die Erfinder auf diese Funktionen gekommen sind ...). Weshalb zwei Schaltkreise? - Gemäß Abbildung 4.8 ist es erforderlich, Signale direkt oder negiert anzuschließen. Das

Zwischenschalten von Negatoren wäre natürlich in diesem Anwendungsfall besonders unschön. Deshalb liefert man zwei Schaltkreise mit gleichem Funktionsausdruck, aber verschiedener Polarität der Signale - die eine Schaltfunktion ist die Negation der anderen. “Programmieren” bedeutet hier: (1) Schaltkreis aussuchen, (2) Signale und Festwerte entsprechend anschließen.

c) typische Einsatzfälle. Wir beschränken uns hier auf Gatter mit 2 Eingängen und auf den Schaltkreis NC7SZ57. Beispielsweise wird die Verknüpfung A UND B gebildet, indem man A an Eigang 1 (I_1) und B an Eingang 2 (I_2) anschließt sowie Eingang 3 (I_3) mit der Speisespannung verbindet (also fest auf High legt).

Hinweis:
Der andere Schaltkreis (NC7SZ58) wirkt - bei gleicher Eingangsbeschaltung - folgendermaßen (entsprechend Abbildung 4.9c): NAND, UND mit invertiertem Eingang A, UND mit invertiertem Eingang B, ODER, XOR (Antivalenz).

Binäre Entscheidungsdiagramme
Das Prinzip des binären Entscheidungsdiagramms läßt sich auf an sich naheliegende Weise ausnutzen, um Schaltfunktionen technisch zu verwirklichen. Jeder Knoten des Diagramms entspricht einer 2-zu-1-Auswahl, die von der jeweiligen Variablen gesteuert wird (Abbildung 4.10), die Verschaltung entspricht der Struktur des Diagramms. Je nachdem, wie die Entscheidungen ausfallen, wird letztlich eine Null oder eine Eins zum Ausgang durchgeschaltet.

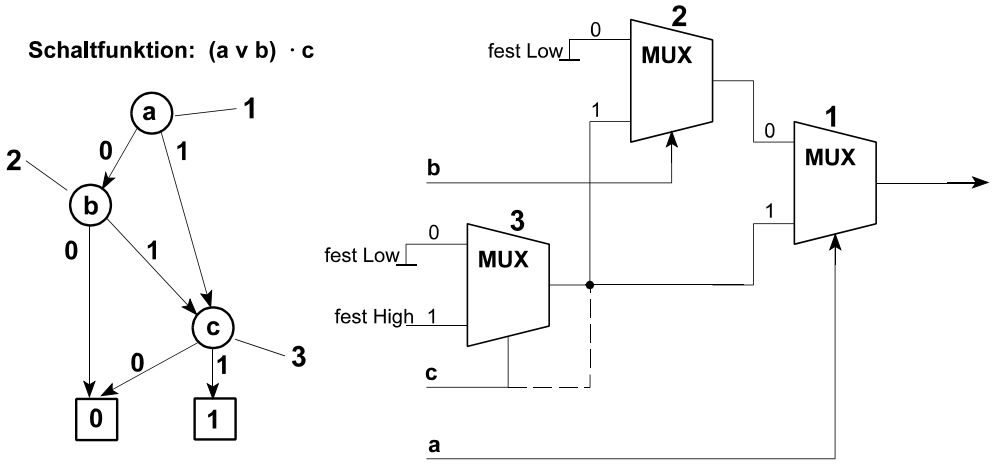


Abbildung 4.10 Das binäre Entscheidungsdiagramm als Grundlage einer Schaltungslösung

Erklärung:
Die 2-zu-1-Auswahl wird hier durch Multiplexer verwirklicht. Betrachten wir das Entscheidungsdiagramm von oben nach unten:

1) ist Variable a = 1, so ist Variable b auszuwerten, ist Variable a = 0, Variable c (der oberste Knoten im Diagramm entspricht dem Multiplexer, der das Ausgangssignal liefert),
2) ist Variable b = 0, so ist der Funktionswert = 0; es ist also eine feste Null durchzuschalten, ist Variable b = 1, so hängt der Funktionswert von der Variablen c ab,

3) hängt der Funktionswert von der Variablen c ab, so ist er Null, wenn c = 0 ist, und er ist Eins, wenn c = 1 ist. (Diesen Multiplexer kann man einsparen und durch Direktanschluß der Variablen c an die Multiplexer 1, 2 ersetzen.)

4.1.3. Sequentielle Schaltungen

Sequentielle Schaltungen enthalten elementare Speichermittel, nämlich Latches oder Flipflops, die teils direkt, teils über kombinatorische Schaltungen miteinander verbunden sind. Will man programmierbare sequentielle Schaltungen anbieten, muß man also eine entsprechende Anzahl von Flipflops oder Latches zusammen mit programmierbaren kombinatorischen Schaltungen auf einem Schaltkreis vorsehen. In einfacheren Schaltkreistypen sind die Speichermittel als einfache Register organisiert oder als - ihrerseits programmierbare - "Makrozellen" (Output Logic Macrocells; OLMCs) (Abbildung 4.11). Einige Beispiele werden wir in den folgenden Abschnitten kennenlernen.

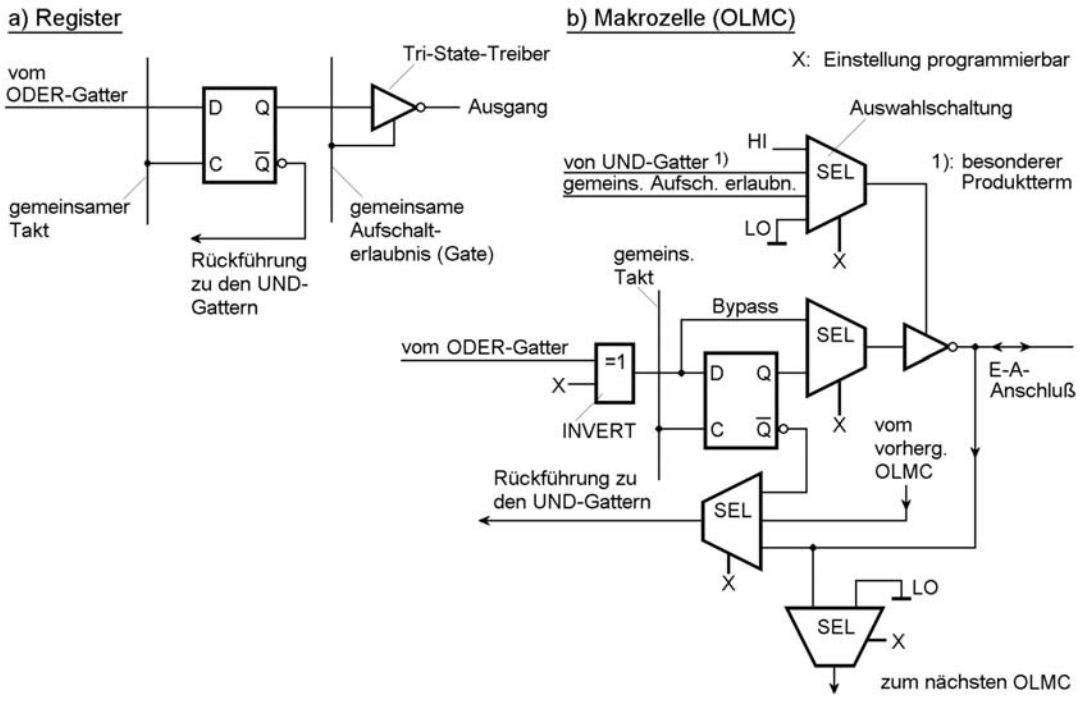


Abbildung 4.11 Flipflopanordnungen in programmierbaren Schaltkreisen (Beispiele, vereinfacht)

Komplexe Makrozellen kann man durch Programmierung für vielfältige Anwendungs-Erfordernisse "zurechtschnitzen". Die Funktionsvielfalt hängt vom jeweiligen Schaltkreistyp ab. Beispielsweise kann man:

- Speichermittel verwenden oder umgehen,
- den Typ des Speichermittels und die Art der Taktsteuerung wählen (z. B. D-Flipflop oder T-Flipflop, Steuerung mit der Low-High- oder mit der High-Low-Flanke),
- die Nutzungsweise des zugehörigen Schaltkreisanschlusses einstellen (z. B. direkter Ausgang, invertierter Ausgang, steuerbarer Tri-State-Ausgang, Eingang),

- interne Rückführungen auf die Kombinatorik schalten,
- ergänzend zur vorgeordneten UND-ODER-Struktur die Eingänge des Flipflops weiteren Signalwandlungen unterziehen (einfachstes Beispiel: Negation über ein steuerbares Antivalenzgatter).

4.1.4. Programmieren

Was ist zu programmieren?
Betrachten wir wieder unsere auf UND-ODER-Strukturen beruhende kombinatorische Hardware (Abbildungen 4.1, 4.6, 4.7): Was ist fest zu verschalten, was ist einstellbar (programmierbar) auszulegen?

1. wir sehen für alle 2^n möglichen Konjunktionen UND-Gatter vor und gestalten die Verbindungen zum ODER-Gatter programmierbar: dies ist das Prinzip des adressierbaren Speichers.
2. wir sehen nur einige UND-Gatter vor, müssen dann allerdings für jeden Gatter- Eingang bestimmen, ob er direkt oder negiert beschaltet werden soll, um tatsächlich jede beliebige Wertekombination aus der Wahrheitstabelle erfassen zu können. Die Verbindungen zwischen UND- und ODER-Gattern können folgendermaßen ausgelegt werden:
   a) die Verbindungen sind ebenfalls programmierbar,
   b) es sind fest an bestimmte UND-Gatter angeschlossene ODER-Gatter vorgesehen.

Kurzdarstellung von Verbindungsstrukturen
Programmierbare Verbindungsstrukturen sind in Matrixform organisiert. Es wäre recht aufwendig - und auch unübersichtlich -, wollte man, um eine bestimmte Organisationsform zu veranschaulichen, alle Verbindungen einzeln zeichnerisch darstellen. Vielmehr ist eine schematische Darstellung gebräuchlich, wobei gleichartige Signalwege zu einer Art Kabelbaum zusammengefaßt und programmierbare Verbindungsstellen besonders gekennzeichnet werden (Abbildung 4.12).

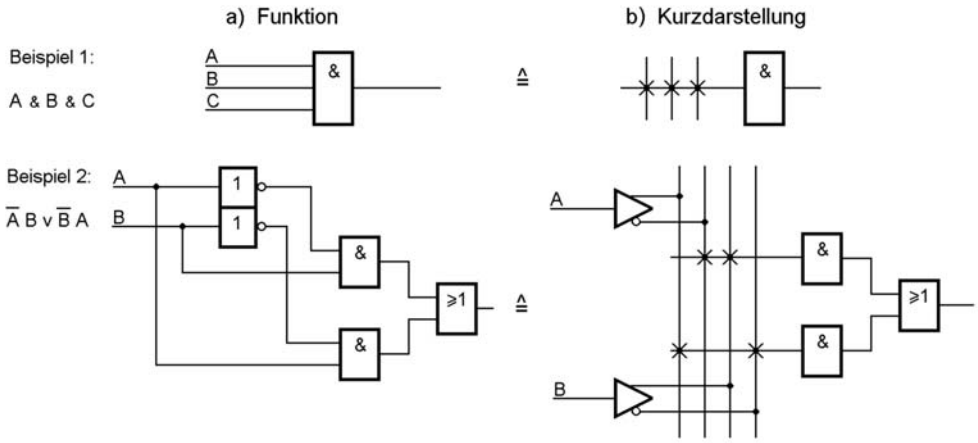


Abbildung 4.12 Kurzdarstellung programmierbarer Verbindungen

4.1.5. Programmierbare Logikschaltkreise: eine Übersicht

Elementare kombinatorische Schaltkreise sind als UND-ODER-Strukturen aufgebaut, die danach unterschieden werden, welche Schaltungsteile von vornherein festgelegt und welche durch Programmierung veränderbar sind (Abbildung 4.13; vgl. auch Abbildung 4.7):

1. UND-Eingänge fest zugeordnet (zudem vollständige Decodierung, also 2^n UNDs bei n Eingängen), ODER programmierbar: adressierbarer *Speicher* (typischerweise ein *ROM* oder PROM),
2. UND-Eingänge programmierbar, ODER programmierbar: *PLA* (Programmable Logic Array),
3. UND-Eingänge programmierbar, ODER-Eingänge fest zugeordnet: *PAL* (Programmable Array Logic).

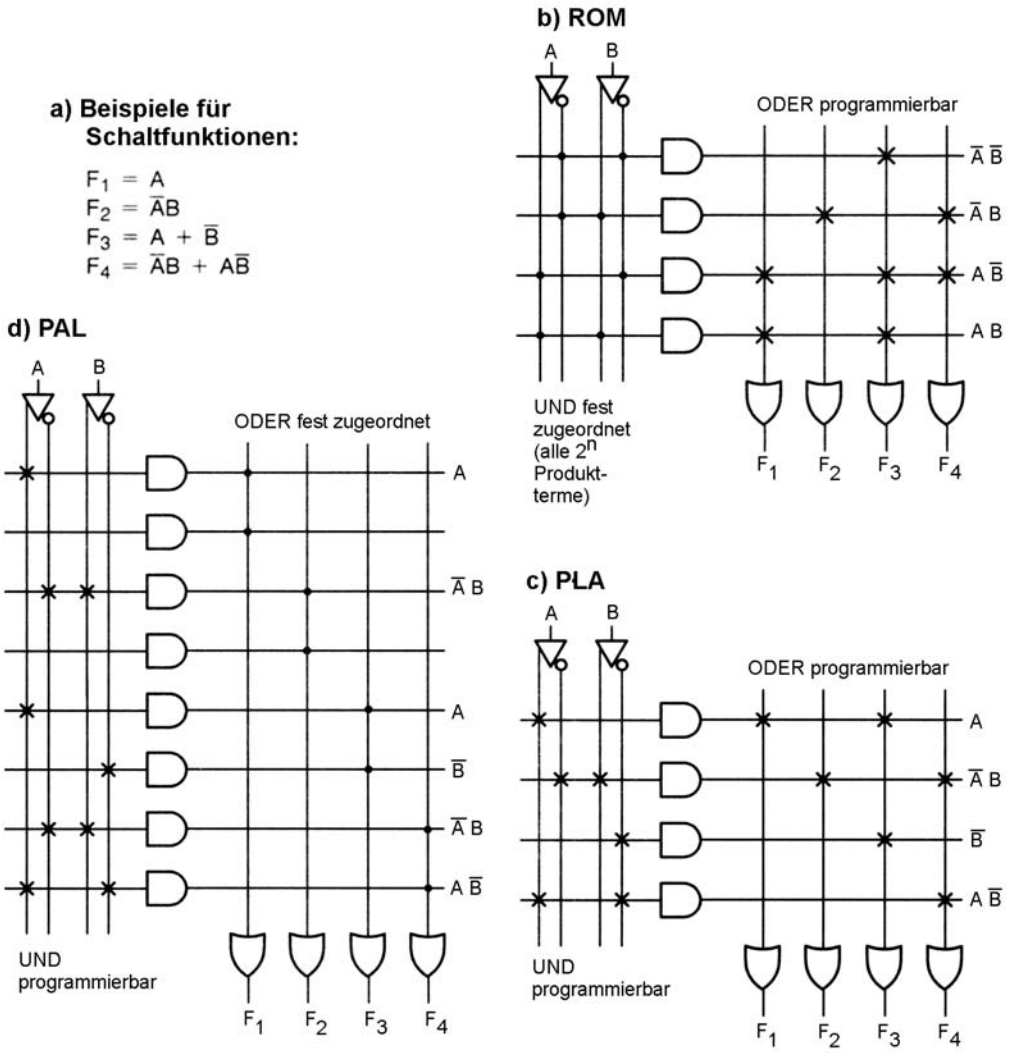


Abbildung 4.13 Elementare programmierbare Logikstrukturen im Vergleich (National Semiconductor). Die altmodischen Schaltsymbole werden auch heute noch gern verwendet

Begriffsbildungen

Die Begriffe sind teils herstellerspezifisch, teils hat sich ein allgemeiner Gebrauch durchgesetzt (obwohl es sich manchmal eigentlich um Warenzeichen handelt).

Programmable Logic Devices (PLDs)

Dies ist der übliche Oberbegriff für programmierbare Logikschaltkreise.

Programmable Logic Arrays (PLAs)

Ein PLA entspricht einer PROM-Struktur mit unvollständiger Adreßdecodierung. Gilt heutzutage als veraltet.

Programmable Array Logic (PALs)
Ein PAL-Schaltkreis enthält eine bestimmte Anzahl von UND-Gattern mit programmierbaren Eingängen. Die UND-Gatter sind fest mit ODER-Gattern verbunden. PAL-Schaltkreise werden zumeist mittels Durchschmelzverfahren (Fusible Link) programmiert.

Erasable Programmable Logic Devices (EPLDs)
Als EPLD bezeichnet man üblicherweise elektrisch (mehrmals) programmierbare und durch UV-Licht löschbare Schaltkreise. Es gibt verschiedene Strukturen , die aber grundsätzlich auf dem Schema UND - ODER - Makrozelle (mit Flipflop) beruhen. Die einfachsten EPLDs könnte man als vielseitigere - und löschbare - PALs bezeichnen; weiterentwickelte EPLDs sind mit GALs und CPLDs vergleichbar (aber nicht immer wirklich kompatibel).

Generic Array Logic (GALs)
Ein GAL-Schaltkreis ist eine elektrisch programmier- und löschbare, um Makrozellen (vgl. Abbildung 4.11) erweiterte PAL-Struktur.

Komplexe programmierbare Logik (Complex Programmable Logic Devices; CPLDs)
Die Bezeichnung ist recht willkürlich. Üblicherweise faßt man unter diesem (oder einem ähnlichen) Begriff all das zusammen, was komplexer ist als eine einfache UND-ODER-Makrozellen-Struktur, aber nicht so komplex wie ein FPGA. (Solche Schaltkreise enthalten, verglichen mit GALs, komplexere Netzwerke und Makrozellen.)

Field Programmable Gate Arrays (FPGAs)
Unter diesem Begriff wollen wir Schaltkreise zusammenfassen, die es gestatten, eine Vielzahl elementarer Teilschaltungen freizügig untereinander zu verbinden, so daß - wenigstens näherungsweise - ein ähnlicher Grad der Schaltungsintegration erreicht werden kann wie bei Nutzung kundenspezifischer Gate-Array-Schaltkreise.

Die Architektur
Der Begriff nimmt sich im Marketing immer gut aus. Er bezeichnet hier die grundsätzliche Auslegung der Verknüpfungen, Zellen und Verbindungsstrukturen.

Wir merken uns:

1. Die grundsätzliche Architektur der PLAs, PALs, EPLDs, GALs und CPLDs entspricht dem Schema Eingänge - UND-Verknüpfungen - ODER-Verknüpfung - Ausgangszelle - Ausgang (Abbildung 4.14).
2. EPLDs, GALs und CPLDs haben vergleichsweise "dicke" UND-ODER-Knoten mit je einem nachgeschalteten Flipflop. Typisch sind 4...8 disjunktiv verknüpfte UNDs mit jeweils 15...40 Eingängen. Diese sog. Makrozellen werden jeweils als Ganzheit verwaltet; es ist nur in beschränktem Umfang möglich, ungenutzte Schaltmittel an andere Zellen gleichsam abzugeben. Die Aufgabe der Entwurfssoftware: den eingegebenen Entwurf auf die gegebene Makrozellenanordnung umzusetzen.
3. GALs sind universelle (und elektrisch löschbare) PALs; die Makrozellen (typisch sind 6...10) sind zumeist direkt an Schaltkreisanschlüsse geführt.

4. CPLDs (auch: manche EPLDs) haben vergleichsweise viele Makrozellen (von etwa 32 an aufwärts bis zu mehreren hundert) sowie interne programmierbare Verbindungsnetzwerke. Nicht alle Makrozellen sind auf Schaltkreisanschlüsse geführt.
5. FPGAs haben keine "dicken" Knoten. Sie beruhen vielmehr auf eher einfachen Zellen, die in eine typischerweise matrixförmige Verbindungsstruktur eingebettet sind (Abbildung 4.15). Die Entwurfssoftware muß entscheiden, was aus der einzelnen Zelle wird - ein Gatter, eine kombinatorische Verknüpfung mit wenigen Eingängen (typisch sind 3...6), ein Multiplexer, ein Flipflop usw.
6. Die Schaltzeiten von Architekturen gemäß Abbildung 4.14 (PLA, PAL...CPLD) sind - wegen des einfachen Aufbaus (UND - ODER- Flipflop) - gut vorhersagbar. Bei FPGAs hingegen ist die Abschätzung von Schaltzeiten schwierig - es kommt sehr darauf an, wie die Entwicklungssoftware die Entwurfsabsicht (z. B. einen Schaltplan) in eine entsprechende Verknüpfung von Zellen umsetzt. Es kann durchaus sein, daß unsere Schaltung bei beispielsweise 100 MHz nicht funktioniert, ungeachtet dessen, daß diese Taktfrequenz in der Werbung immer wieder herausgestellt wird.
7. Nur ganz elementare Schaltkreise (PALs, GALs) sind gleichsam Industriestandards. EPLDs, CPLDs und FPGAs sind herstellerspezifisch. Was u. a. geschehen kann:
   - daß die Baureihe (oder gar der Hersteller) vom Markt verschwindet,
   - daß Hersteller aufgekauft, Fertigungslinien verkauft oder Lizenzen vergeben werden (so daß die Baureihe X womöglich sang- und klanglos verschwunden ist, wir sie aber von einem anderen Anbieter - womöglich unter anderem Handelsnamen - nach wie vor beziehen können).

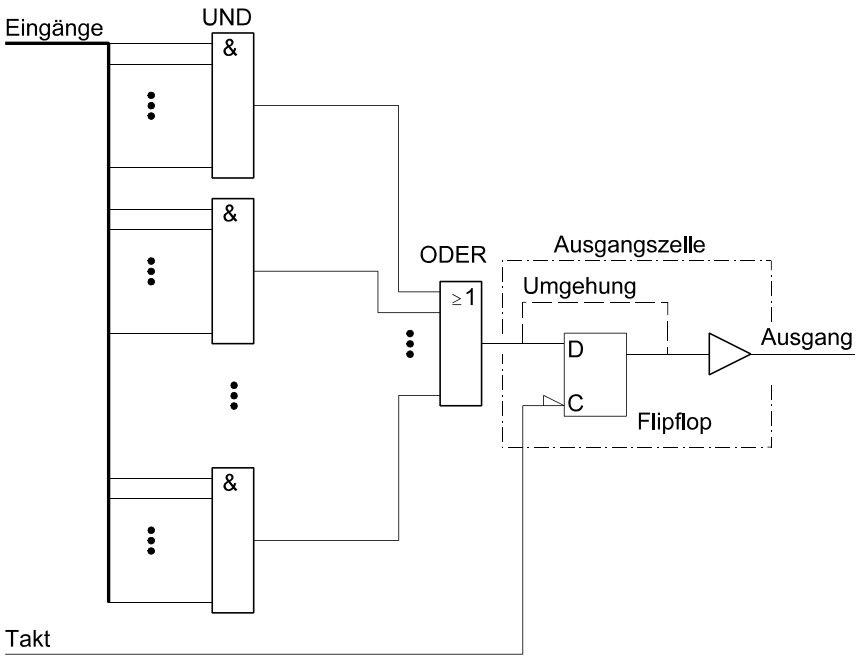


Abbildung 4.14 Die grundsätzliche Architektur eines programmierbaren Logikschaltkreises (PLA, PAL, EPLD, GAL, CPLD. Die Schaltkreise enthalten mehrere...viele derartige Anordnungen (Makrozellen)

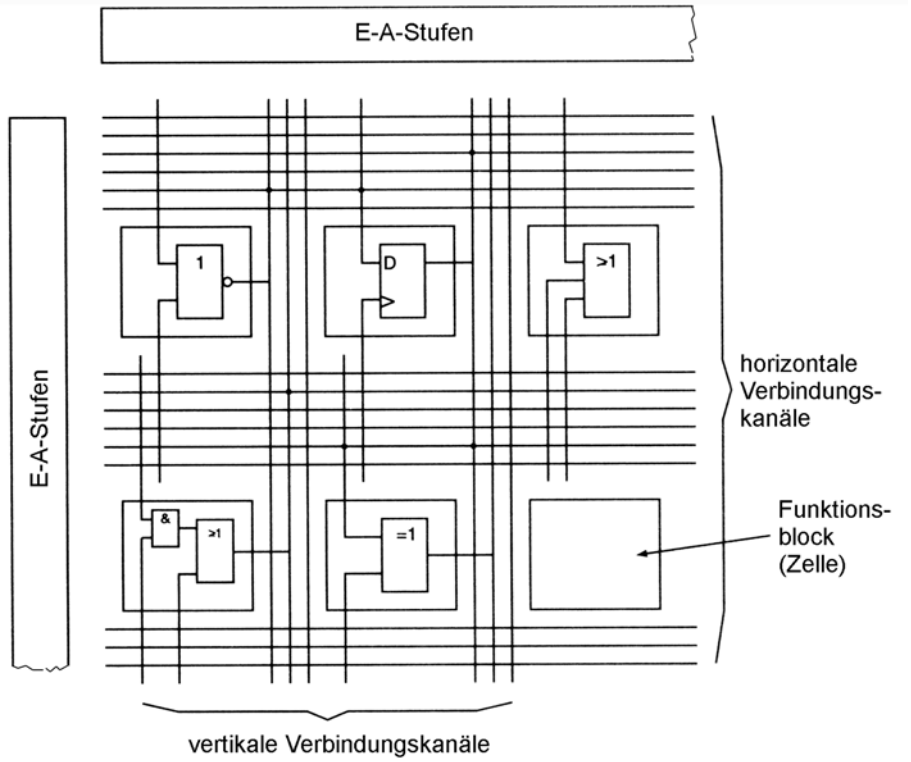


Abbildung 4.15 Die grundsätzliche Architektur eines FPGAs (Texas Instruments)

PLA oder PAL?
Weshalb beruht die programmierbare Logik - bis hin zum CPLD - vor allem auf PAL-Strukturen (UND programmierbar, ODER fest)? Eine Schaltung, deren ODER-Verknüpfungen ebenfalls programmierbar sind (also ein PLA), ist doch viel universeller? - Ja, aber:

- es ist teurer (die Programmiervorkehrungen für die ODER-Verknüpfungen kosten Siliziumfläche),
- es ist langsamer (programmierbare Signalwege haben eine größere Durchlaufverzögerung als Festverbindungen).

Technologien
Programmierbare Logik wird in den Technologien (Schottky-) TTL, CMOS und ECL gefertigt. Elektrische Programmierbarkeit durch Ladungsspeicherung ist nur in CMOS zu verwirklichen. TTL- und ECL-Schaltungen (in moderner Hardware eher selten) werden nach dem Durchschmelzprinzip (Fusible Link) programmiert.

Was wird eigentlich programmiert?
Wie die Programmiervorkehrungen im einzelnen ausgelegt sind - darüber sind die Hersteller zumeist nicht sehr mitteilsam. Als Anwender (= Schaltungsentwickler) braucht man solche Einzelheiten auch gar nicht. Wir wollen uns deshalb mit einem Überblick über einige elementare Grundlagen begnügen (Abbildungen 4.16 bis 4.18).

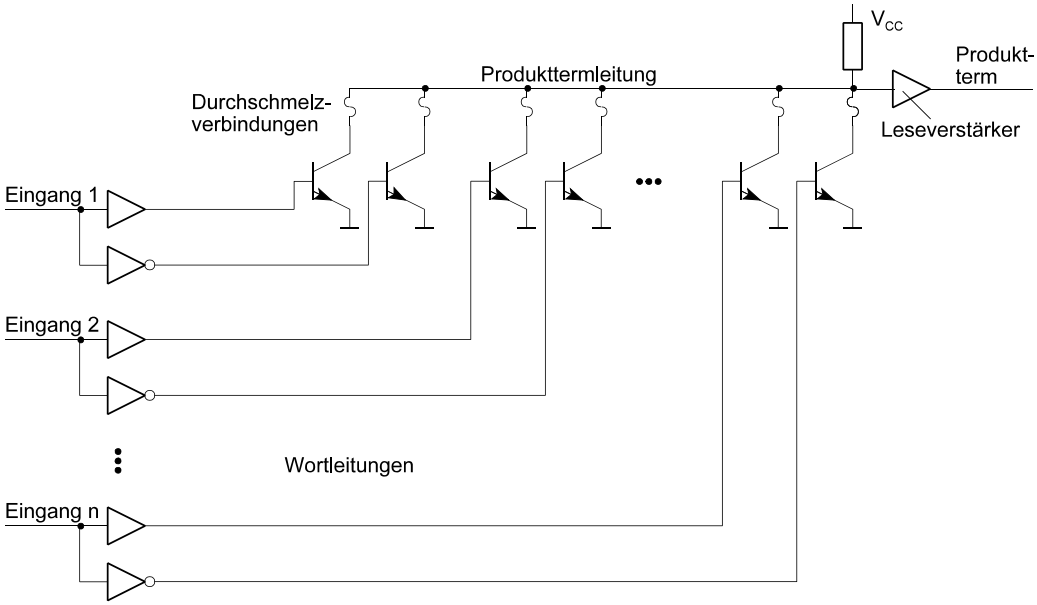


Abbildung 4.16 Eine programmierbare UND-Verknüpfung (Produktterm). Prinzipschaltung, vereinfacht

Erklärung:
In herkömmlichen (bipolaren) PALs (TTL, ECL) ist die Produkttermbildung ähnlich aufgebaut wie ein PROM. Den Eingängen sind Pufferstufen nachgeschaltet, die sowohl die direkte als auch die negierte Eingangsbelegung auf jeweils eine sog. Wortleitung*) geben. Nehmen wir an, alle Durchschmelzverbindungen (Fusible Links) seien noch vorhanden. Dann wird jede Erregung einer Wortleitung den betreffenden Transistor durchschalten, so daß die Produkttermleitung auf Low-Pegel gezogen wird. Transistoren, Produkttermleitung und Widerstand bilden ein sog. Wired NOR - ein High an der Basis eines Transistors genügt, um die Produkttermleitung auf Low zu schalten. Wie kommt nun eine UND-Verknüpfung zustande? - Indem die Entwurfssoftware die DeMorganschen Regeln anwendet:

$$a \cdot b \cdot c = \overline{\overline{a} \vee \overline{b} \vee \overline{c}}.$$

Beim Programmieren sind die Durchschmelzverbindungen jener Wortleitungen, deren Signale in die NOR-Verknüpfung eingehen sollen, zu erhalten. Die anderen sind aufzutrennen (durch Anlegen einer entsprechend hohen Programmierspannung).

*): in Entsprechung zum PROM.

Hinweis:
Eine Produkttermleitung stellt - infolge ihrer Länge und der vielen angeschlossenen Transistoren - eine vergleichsweise große kapazitive Last dar. Damit die Schaltzeiten trotzdem gering bleiben, arbeitet man mit sehr geringen Spannungshüben (bis hinab zu 100 mV). Deshalb muß ein Leseverstärker nachgeschaltet werden.

Denksportaufgabe:
Es sind nur die ersten 4 Eingänge belegt. Wir wollen folgende Verknüpfung realisieren: $1 \cdot 2 \cdot \overline{3} \cdot 4$. Welche Verbindungen sind zu trennen, welche zu erhalten?

Wir müssen folgende NOR-Verknüpfung aufbauen: $\overline{1 \vee \overline{2} \vee 3 \vee \overline{4}}$. Demgemäß sind die Durchschmelzverbindungen der negierten Eingänge 1, 2, 4 und des nichtnegierten Eingangs 3 zu erhalten. Alle anderen sind zu trennen.

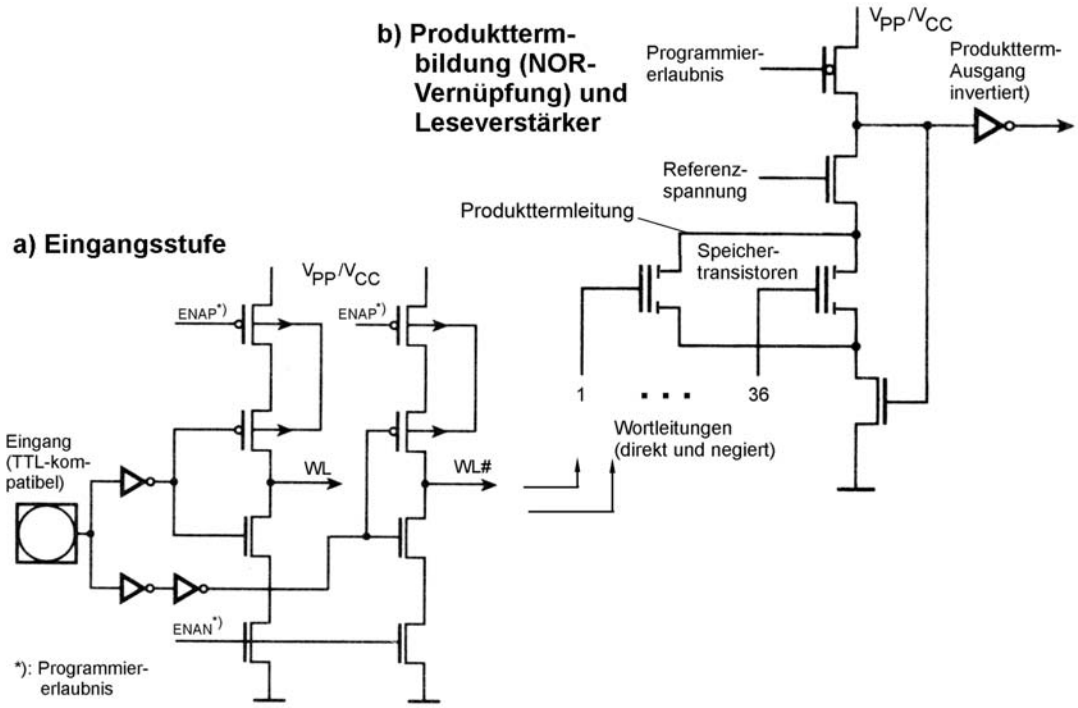


Abbildung 4.17 Eingangsstufe mit nachgeschalteter UND-Verknüpfung in einem EPLD-Schaltkreis (Intel)

Es handelt sich um einen Ausschnitt aus einem EPLD-Schaltkreis. Die Eingangsstufe (a) erregt zwei Wortleitungen (direkt: WL, invertiert: WL#). Zur Produkttermbildung (b) dienen hier EPROM-Speicherzellen, also Feldeffekttransistoren mit einem zusätzlichen isolierten Gate (Speicher-Gate). Sie sind parallelgeschaltet und wirken somit als NOR-Verknüpfung. Trägt das Speicher-Gate keine Ladung, so bewirkt ein High-Pegel auf der angeschlossenen Wortleitung, daß die Source-Drain-Strecke leitend wird. Die Leseverstärkeranordnung schaltet somit eingangsseitig auf Low. Ist das Speicher-Gate geladen, kann sich die Erregung der Wortleitung nicht auswirken. Der Transistoranordnung ist ein (hier invertierend wirkender) Verstärker nachgeschaltet, der ausreichend hohe Pegel für die nachgeordnete ODER-Verknüpfung liefert.

Die weiteren Transistoren in der Abbildung dienen zum Programmieren. Die Beschaltung entspricht der eines EPROMs: Substrat und Source an Masse, positive Spannungen an Auswahl-Gate und Drain (um Ladungsträger auf das Speicher-Gate zu schaffen). Vor dem Programmieren ist der Schaltkreis mittels UV-Licht zu löschen.

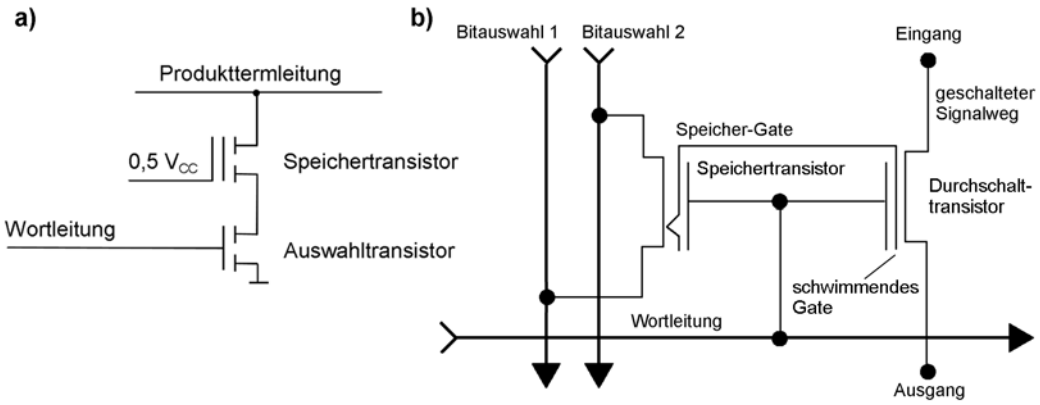


Abbildung 4.18 Löschbare Verknüpfungen (Beispiele)

Erklärung:

a) einfache EEPROM-Zelle. Manche Flash-Zellen sind ähnlich aufgebaut (Zweitransistorkonfiguration). Der Speichertransistor tritt hier an die Stelle der Durchschmelzverbindung von Abbildung 4.16. Ist das Speicher-Gate geladen, so ist der Stromweg freigegeben, ist es gelöscht, ist der Stromweg gesperrt.

b) Flash-Schalterelement (Actel). Dem eigentlichen Speichertransistor ist ein weiterer Transistor mit schwimmendem Gate (Floating Gate) parallelgeschaltet, der den Datenweg sperrt oder freigibt. Die Wortleitung und die beiden Bitauswahlleitungen dienen zum Programmieren. Der Vorteil dieser Konfiguration: Programmierung und Datenwegsteuerung sind voneinander getrennt - und können somit jeweils für sich optimal ausgelegt werden (wir können uns das Ganze als eine Flash-Speichermatrix vorstellen, in der jede Speicherzelle mit einem Schalttransistor verbunden ist). RAM-Zellen sind ähnlich aufgebaut (SRAM-Speichermatrix mit angeschlossenen Schalttransistoren).

Wichtige Kennwerte

Die genaue Dokumentation programmierbarer Schaltkreise füllt umfangreiche Datenbücher. Der funktionellen Komplexität entsprechen naturgemäß vielfältige statische und dynamische Kennwerte. Wenn Sie es genau wissen wollen, bleibt Ihnen der Blick ins Datenmaterial (Internet) nicht erspart. Für eine überschlägige Orientierung hingegen sind zwei Kennwerte von besonderer Bedeutung:

1. die Durchlaufverzögerungszeit der Kombinatorik (gemessen vom Schaltkreiseingang bis zum Ausgang bzw. bis zum Eingang des anzusteuernden Flipflops),

2. die maximale Taktfrequenz (mit der eine sequentielle Schaltung betrieben werden kann). Der Wert hängt von der Betriebsweise ab. Deshalb werden üblicherweise zwei verschiedene Werte angegeben: (1) die Taktfrequenz bei Nutzung schaltkreisinterner Rückführungen, (2) die Taktfrequenz ohne Nutzung dieser Rückführungen (zumeist als Flow-Through- oder als Pipeline-Modus bezeichnet).

4.2. Universelle und programmierbare Logik mit Standardschaltkreisen

Multiplexer und Decoder
Insbesondere Multiplexer, manchmal aber auch Decoder, sind gern als "Logikersatz" verwendet worden. Obwohl seit dem Erscheinen der GALs an sich veraltet, können solche Lösungen aber auch in neuerer Hardware noch vorkommen (ein naheliegender Grund: man muß nicht programmieren, braucht also weder Programmiergerät noch Entwicklungssoftware). Abbildung 4.19 veranschaulicht das Prinzip am Beispiel eines binären Volladdierers.

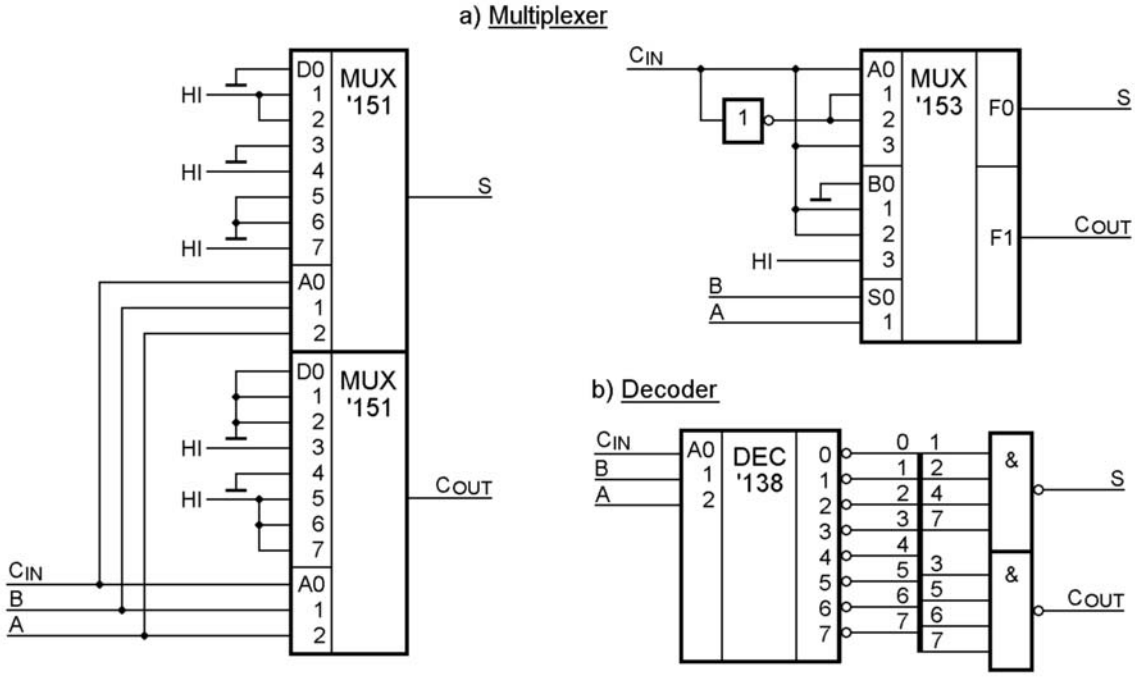


Abbildung 4.19 Realisierung eines binären Volladdierers

Rekonstruktion der Wahrheitstabelle
Solche Schaltungen sind zumeist ohne besondere Erläuterungen im Schaltplan dargestellt. Um die Wirkungsweise zu verstehen, müssen wir die Wahrheitstabelle aus dem Schaltplan rekonstruieren. Kapitel 8 enthält ein entsprechendes Übungsbeispiel (Aufgabe 10).

Adressierbare Speicher
Adressierbare Speicher haben den Vorteil, daß man mit ihnen jede beliebige Schaltfunktion (in den Grenzen der gegebenen Adreßeingänge) verwirklichen kann - es ist vollkommen gleichgültig, wie kompliziert die kombinatorische Zuordnung ist. Wegen der einfachen Anwendung werden vorzugsweise ROMs oder PROMs der verschiedensten Technologien eingesetzt. Bevorzugte Einsatzgebiete sind:

- Zuordner in State Machines,
- Tabellen zur Codewandlung (z. B. von EBCDIC nach ASCII und umgekehrt),

- beliebige andere Zuordnungstabellen, die in der Hardware benötigt werden (Beispiele: die Additions-Subtraktions-Tabelle oder auch das "kleine Einmaleins" des Dezimalrechnens, binär codierte Wertetafeln zum Erzeugen von an sich beliebigen Signalverläufen u. ä.),
- Zeichengeneratoren in Videoadaptern, Druckern usw.,
- beliebige Funktionen der "Glue Logic" einschließlich der Adreßdecodierung (so war im "klassischen" PC/AT die Adreßdecodierung auf dem Motherboard teilweise mit Schottky-TTL-PROMs realisiert).

Abbildung 4.20 zeigt einige Beispiele.

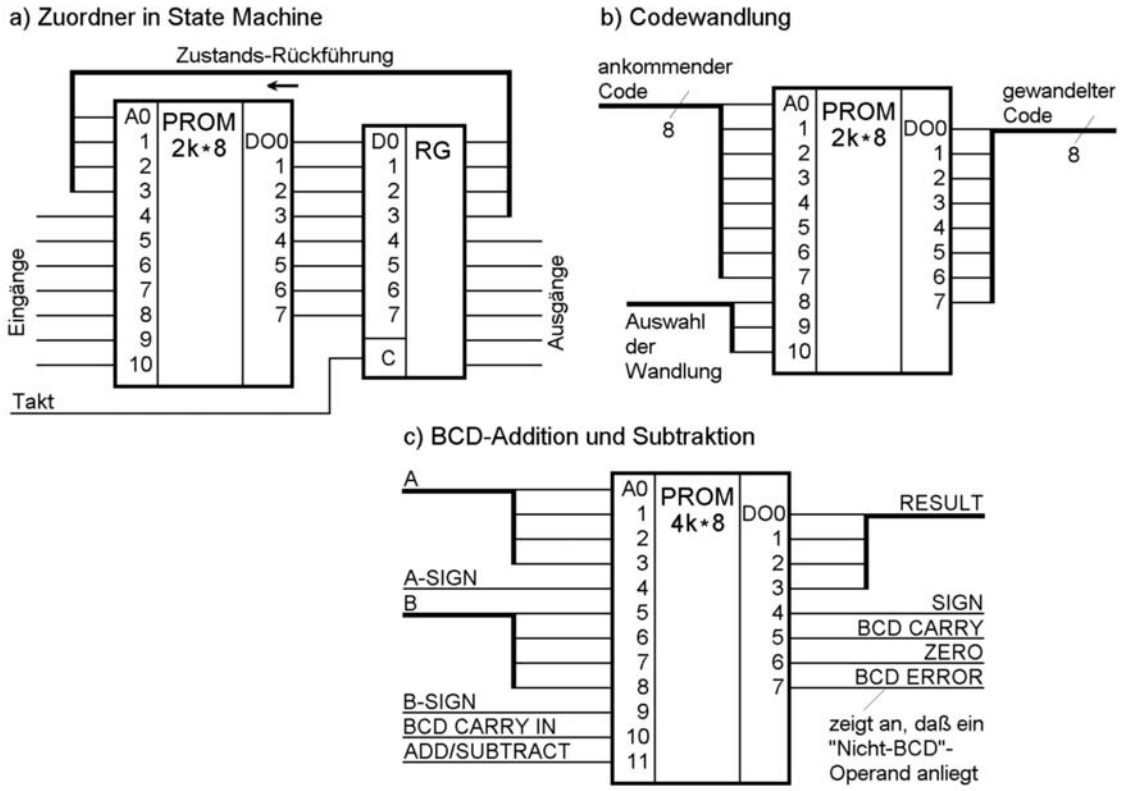


Abbildung 4.20 PROMs als universelle Logikzuordner

4.3. PALs (Programmable Array Logic)

Kombinatorische PALs

PALs haben programmierbare UND-Verknüpfungen, denen ODER-Verknüpfungen fest nachgeschaltet sind (Abbildung 4.21). Der einzelne Schaltkreis ist demzufolge vergleichsweise unflexibel (es ist eben nicht "alles" programmierbar). Zum Ausgleich gibt es eine beachtliche Typenvielfalt. Bessere Entwicklungssoftware ist in der Lage, aus dem Sortiment jene Typen auszuwählen, die zur Realisierung des jeweiligen Entwurfs geeignet sind.

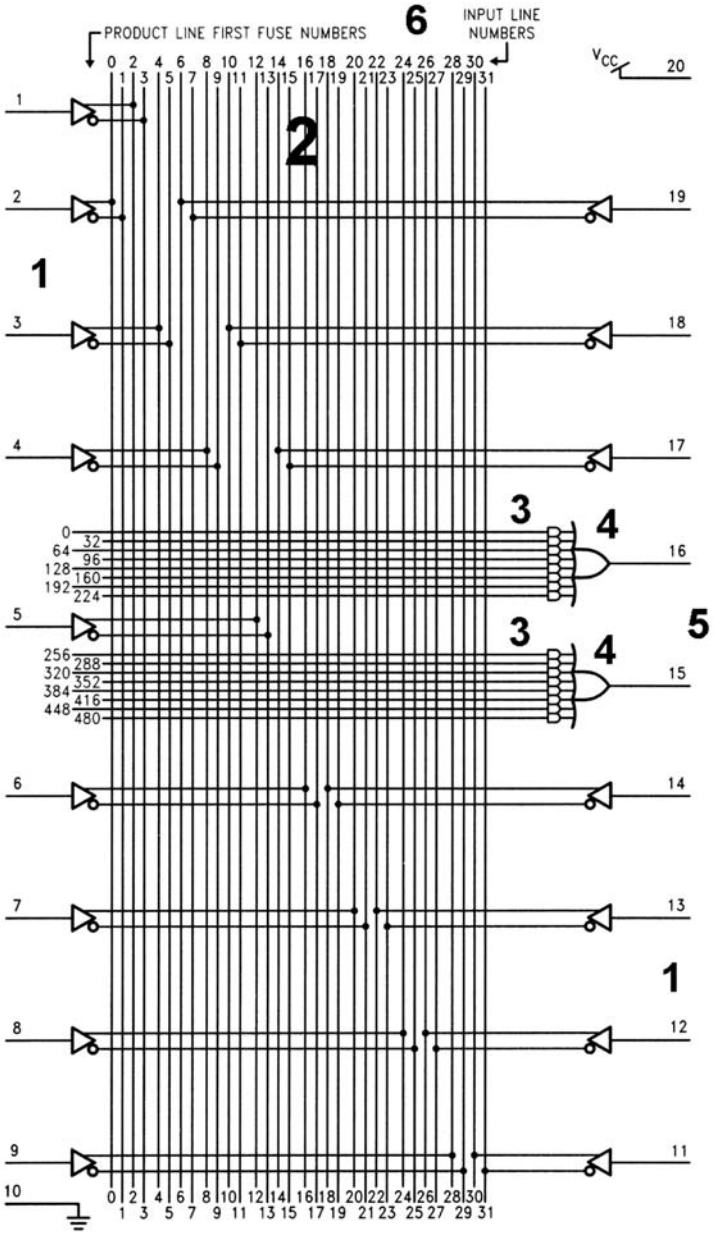


Abbildung 4.21 Beispiel eines PAL-Schaltkreises: PAL 16H2

Erklärung zu Abbildung 4.21:
1 - Eingangspuffer; 2 - Wortleitungen; 3 - UND-Verknüpfungen (Produkttermleitungen mit nachgeschaltetem Leseverstärker); 4 - ODER-Verknüpfung; 5 - Ausgänge; 6 - Adreßangaben für die Programmierpositionen (Fuse Numbers). Der Schaltkreis hat 16 Eingänge und 2 Ausgänge. Jedem der ODER-Verknüpfungen 4 sind acht 32-fach-UND-Verknüpfungen 3 vorgeschaltet, die gemäß Abbildung 4.16 ausgebildet sind. Die Eingangspuffer 1 liefern wahre und invertierte Signale an die Wortleitungen 2, die über Durchschmelzverbindungen mit den Produkttermleitungen 3 verknüpft sind.

Jede der beiden UND-ODER-Anordungen ermöglicht es somit, eine Boolesche Gleichung zu implementieren, die maximal 16 Variable hat und deren DNF aus höchstens 8 UND-Termen besteht (wobei es gleichgültig ist, wieviele Variable der einzelne UND-Term enthält).

Programmierung
Jede zu programmierende Position wird durch eine Adreßangabe (Fuse Number) gekennzeichnet. Diese sind nach JEDEC standardisiert. Die Entwicklungssoftware erzeugt entsprechende Dateien (sog. JEDEC Files), die zum Programmiergerät übertragen werden.

Ausgänge
Manche PALs haben zweiwertige Logikausgänge (bei "H"-Typen direkt, bei "L"-Typen invertiert wirkend), manche haben Tri-State-Ausgänge. Die Aufschalterlaubnissignale werden mit programmierbaren UND-Verknüpfungen gebildet. Einige (gelegentlich auch alle) Ausgänge sind in die UND-Gatter-Matrix zurückgeführt (Feedback) und können so in die Bildung von Produkttermen mit einbezogen werden.

Kennwerte
Durchlaufverzögerung: 7...35 ns (Schottky TTL); 2...4 ns (ECL).

Sequentielle (Registered) PALs
Diese PALs enthalten Flipflops, die den Ausgängen vorgeschaltet sind. Die Flipflops werden durch einen gemeinsamen Takt gesteuert und bilden somit praktisch ein Register. Auch hier gibt es eine große Typenvielfalt. Abbildung 4.22 zeigt nähere Einzelheiten einer solchen PAL-Struktur.

Rückführungen
Die Flipflopausgänge sind auf die UND-Gatter-Matrix zurückgeführt und können so in die Bildung von Produkttermen einbezogen werden. Das ermöglicht es, in solchen PALs auch Zähler, Schieberegister usw. zu realisieren.

Asynchrone Flipflops
Es gibt PAL-Typen, die asynchron setz- bzw. rücksetzbare Flipflops enthalten. Abbildung 4.23 zeigt ein Beispiel (es ist einer der 8 Ausgänge genauer dargestellt).

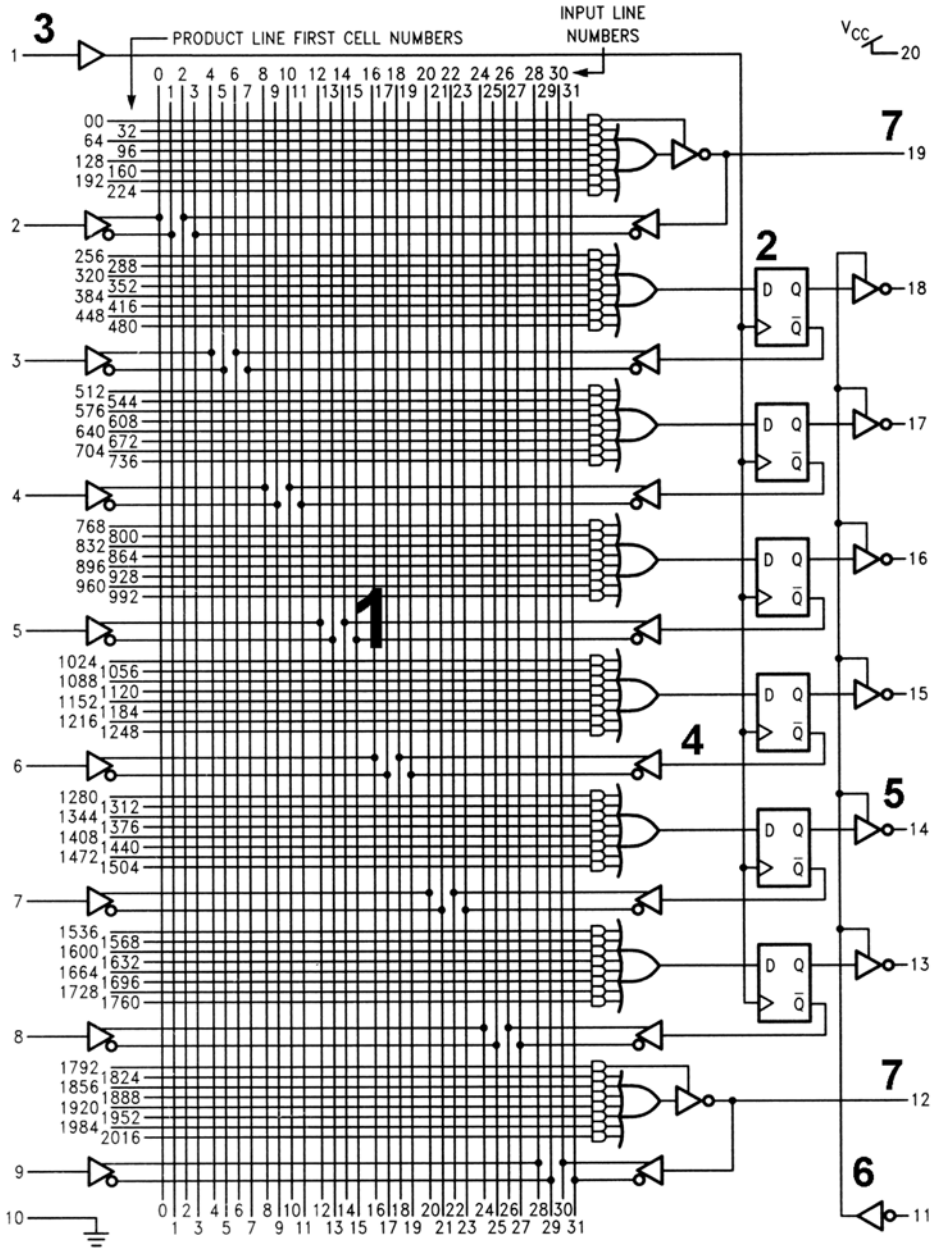


Abbildung 4.22 Ein PAL-Schaltkreis mit Flipflops: PAL 16R8

Erklärung:
1 - UND-ODER-Struktur; 2 - Flipflop; 3 - gemeinsamer Takt; 4 - Rückführung des Flipflop-Ausgangs; 5- Ausgangstreiber (Tri State); 6 - gemeinsames Aufschalterlaubnissignal (Output Enable); 7 - kombinatorische Ausgangssignale.

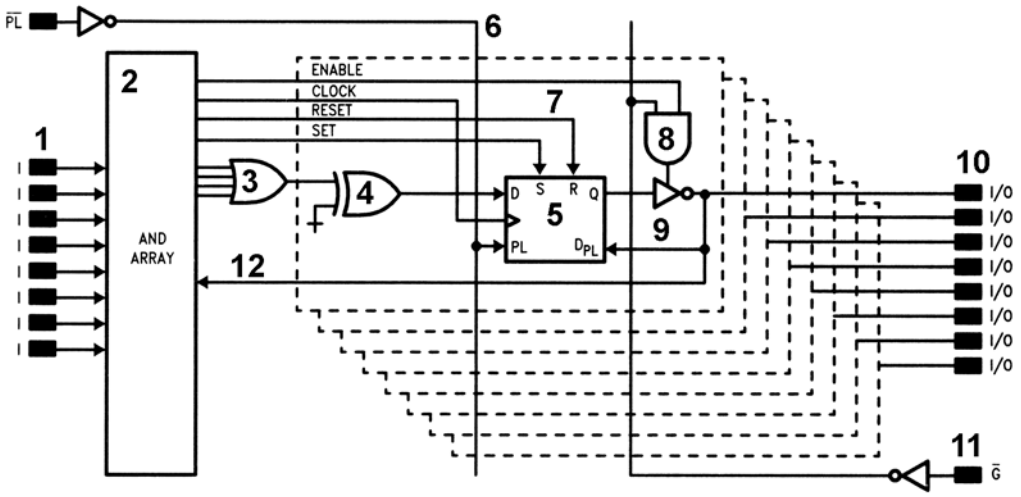


Abbildung 4.23 PAL 16RA8

Erklärung:
1 - Eingänge; 2 - UND-Anordnung; 3 - ODER-Verknüpfung; 4 - XOR-Gatter zum Programmieren der ausgangsseitigen Polarität (aktiv High oder aktiv Low); 5 - Flipflop; 6 - gemeinsamer Ladetakt (wirkt asynchron durch Setzen/Rücksetzen des Flipflops); 7 - einzeln über UND-Verknüpfungen gebildete Steuersignale (Setzen, Rücksetzen, Takt); 8 - Erlaubnissignalbildung für Tri-State-Ausgang; 9 - Anschluß kann als einzelner asynchron wirkender Ladetakteingang genutzt werden; 10 - E-A-Anschlüsse; 11 - gemeinsamer Erlaubnissignaleingang; 12 - Rückführung vom Anschluß 10 auf die UND-Anordnung (E-A-Anschluß als Eingang genutzt).

Das Flipflop und seine Zusatzbeschaltung bilden hier eine Art Makrozelle (Output Logic Macrocell OLMC), die sehr vielseitig programmiert werden kann. So kann man durch passendes Programmieren das Flipflop umgehen (Bypass) und kommt so zu einem rein kombinatorisch angesteuerten Ausgang. Auch wird der Takt über besondere Produktterme aus der UND-Anordnung gebildet; es läßt sich also für jedes Flipflop eine gesonderte Taktsteuerung vorsehen.

Kennwerte von sequentiellen PALs
Schottky-TTL: Durchlaufverzögerung: 25 ns, Taktfrequenz bei Nutzung von Rückführungen: 37 MHz, Taktfrequenz ohne Nutzung von Rückführungen: 50 MHz. ECL: Durchlaufverzögerung: 4...6 ns, Taktfrequenz: 117...200 MHz.

4.4. GALs (Generic Array Logic)

GALs sind gleichsam universelle PALs, die elektrisch programmier- und löschbar sind. Sie enthalten EEPROM- oder Flash-Speicherzellen. Die typische GAL-Grundstruktur: programmierbare UND-Verknüpfungen - fest nachgeschaltete ODER-Verknüpfungen - programmierbare Funktionsblöcke (Output Logic Macrocells OLMCs) - Ausgänge. Auf dieser Grundlage kann man sowohl kombinatorische als auch sequentielle Schaltungen verwirklichen,

und man kommt mit einem - im Vergleich zu PALs - geringeren Schaltkreissortiment aus. Die Abbildungen 4.24 bis 4.28 zeigen nähere Einzelheiten. Abgesehen von der Vielseitigkeit und vom anderen Programmierverfahren bieten die GALs dem Schaltungsentwerfer die gleichen Möglichkeiten wie entsprechend ausgewählte PALs.

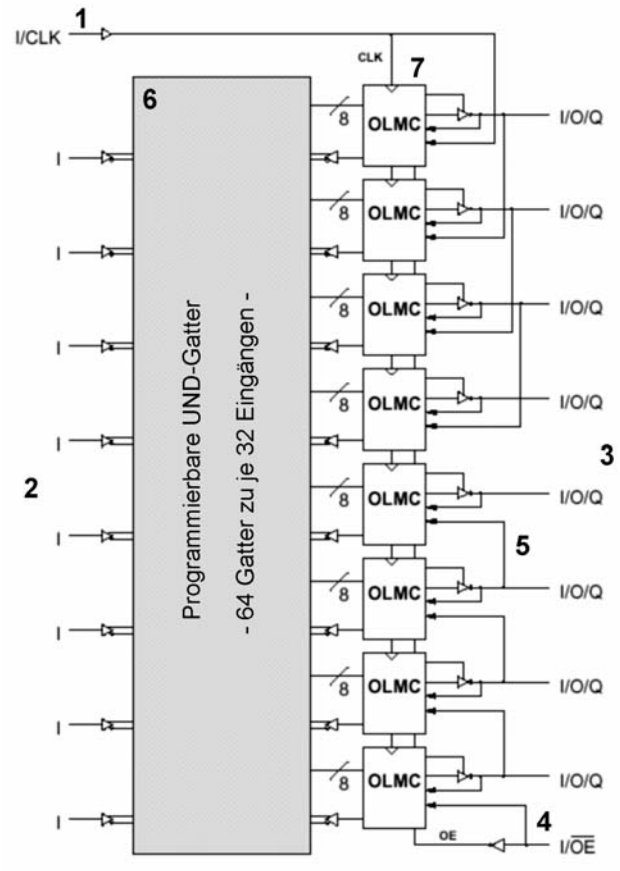


Abbildung 4.24 GAL 16V8 (Lattice)

Erklärung:
1 - gemeinsamer Takteingang; 2 - Eingänge; 3 - E-A-Anschlüsse*); 4 - gemeinsamer Erlaubnissignaleingang; 5 - Weiterschaltung zur nächsten Makrozelle**); 6 - UND-Anordnung; 7 - Makrozellen. Der Schaltkreis hat 8 Eingänge und 8 E-A-Anschlüsse. Jedem E-A-Anschluß ist eine Makrozelle (Abbildung 4.25) vorgeschaltet. Die UND-Verknüpfungen betreffen alle 16 Signale (8 von den Eingängen 2 und 8 Rückführungen aus den Makrozellen 7). Jede Makrozelle hat ein vorgeschaltetes Achtfach-ODER. Die UND-Anordnung 6 enthält somit $8 \cdot 8 = 64$ UNDs zu 32 Eingängen (= 16 Variable direkt und negiert). Die UND-Verknüpfungen sind ähnlich Abbildung 4.16 aufgebaut, enthalten aber EEPROM-Zellen ähnlich Abbildung 4.18a.

*): wahlweise als Eingänge, als Ausgänge oder bidirektional.

**): zur besseren Ausnutzung der E-A-Anschlüsse. Es kann sein, daß eine Zelle “ihren” Anschluß gar nicht verwendet. Er kann dann trotzdem als Eingang genutzt werden, indem er über die Rückführung in der benachbarte Zelle an die UND-Anordnung angeschlossen wird.

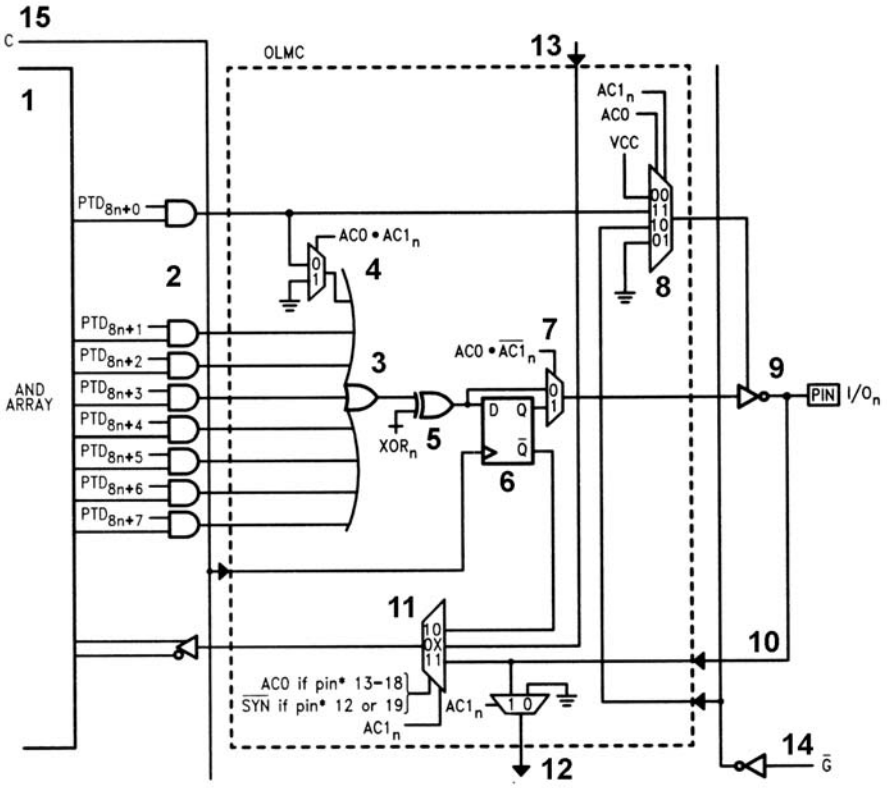


Abbildung 4.25 Die Makrozelle (Output Logic Macrocell OLMC) eines GAL 16V8

Erklärung:
1 - UND-Anordnung; 2 - Produkttermabschaltung*); 3 - das 8-fach-ODER; 4 - programmiertes Abzweigen eines Produktterms**); 5 - XOR-Gatter zum Programmieren der ausgangsseitigen Polarität (aktiv High oder aktiv Low); 6 - D-Flipflop; 7- programmierbare Umgehung (Bypass); 8 - Wahl des Aufschalterlaubnissignals; 9 - Tri-State-Ausgangstreiber; 10 - Eingangsleitung vom E-A-Anschluß; 11 - Wahl der Rückführung auf die UND-Anordnung; 12 - Weiterreichen des Eingangssignals zur nächsten Makrozelle; 12 - Eingangssignal von vorhergehender Makrozelle; 14 - gemeinsamer Erlaubnissignaleingang; 15 - gemeinsamer Takteingang.

*): Trick: PTD = Product Term Disable. Produkttermleitungen, die gar nicht genutzt werden, schaltet man über eine weitere programmierbare UND-Verknüpfung ganz ab (Stromsparen, Verringerung der Störbeeinflussung).

**): Trick: einer der 8 Produktterme kann genutzt werden, um das Aufschalterlaubnissignal für den Tri-State-Ausgang zu bilden. Wird das so programmiert, stehen der “eigentlichen” Verknüpfung nur noch 7 Produktterme zur Verfügung.

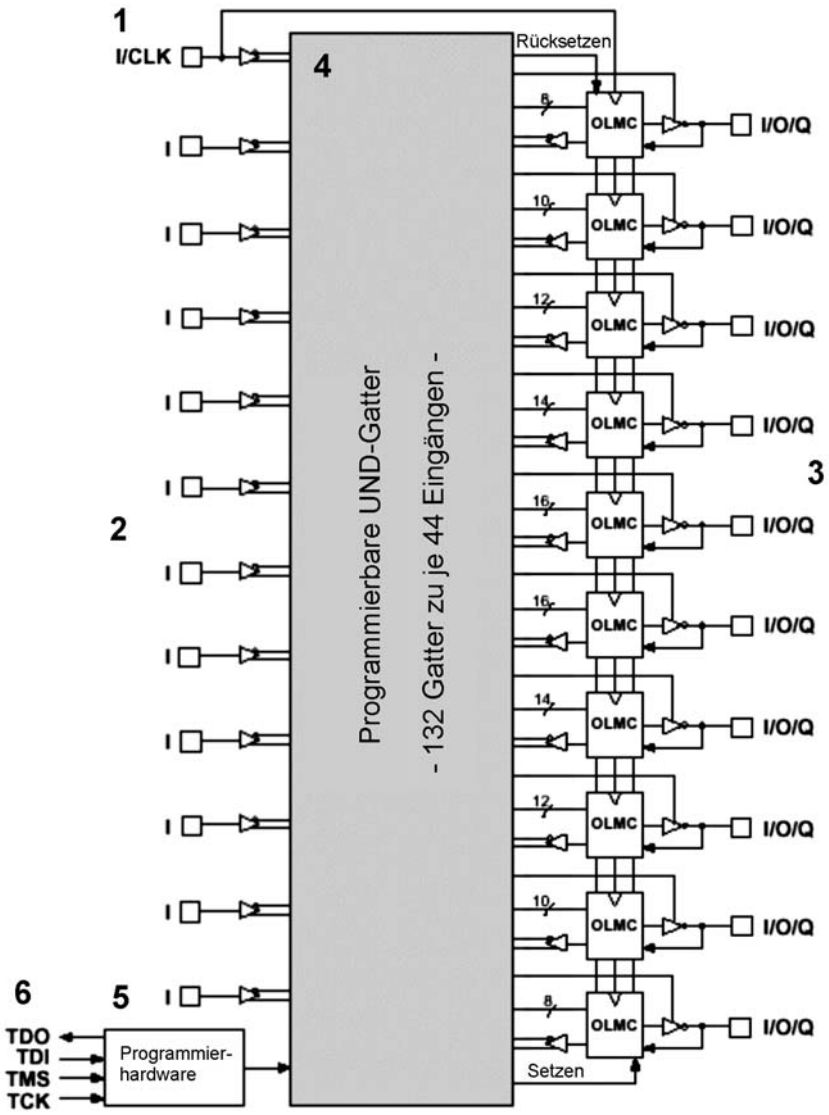


Abbildung 4.26 ispGAL22V10 (Lattice)

Erklärung:
Ebenso wie die 16V8 ist die 22V10 ein Allerweltstyp, der in riesigen Stückzahlen zum Einsatz kommt. Hier zeigen wir eine Ausführung, die sich in eingebautem Zustand programmieren läßt (isp = in system programmable). 1 - gemeinsamer Takteingang*); 2 - Eingänge; 3 - E-A-Anschlüsse; 4 - UND-Anordnung; 5 - Programmierhardware; 6 - die zusätzlichen Programmieranschlüsse (hier: eine sog. Boundary-Scan- oder JTAG-Schnittstelle).

*): auch als gewöhnlicher Eingang nutzbar.

Der Schaltkreis hat 12 Eingänge und 10 E-A-Anschlüsse mit vorgeschalteten Makrozellen (Abbildung 4.27). Die Eingänge und die Rückführungen aus den Makrozellen können an insgesamt 132 UND-Verknüpfungen angeschlossen werden. Jeweils zwei Makrozellen haben ODER-Verknüpfungen mit 8, 10, 12, 14 und 16 Eingängen.

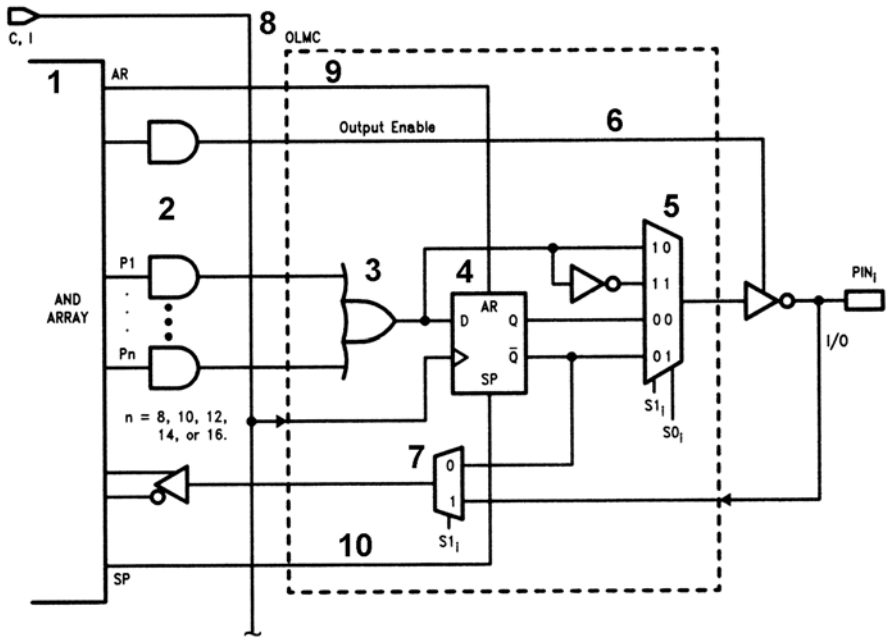


Abbildung 4.27 Die Makrozelle (Output Logic Macrocell OLMC) eines GAL 22V10

Erklärung:
1 - UND-Anordnung; 2 - Produkttermabschaltung; 3 - ODER-Verknüpfung (von 8, 10, 12, 14 oder 16 Produkttermen); 4 - D-Flipflop; 5- programmierbare Umgehung (Bypass) und Polaritätswahl (Abbildung 4.28); 6 - Produktterm für Aufschalterlaubnissignal; 7 - Wahl der Rückführung auf die UND-Anordnung; 8 - gemeinsamer Takteingang; 9 - Produktterm zum asynchronen Rücksetzen; 10 - Produktterm zum synchronen Setzen (9 und 10 sind allen Makrozellen gemeinsam). Offensichtlich ist 22V10 einfacher ausgelegt als 16V8, hat aber mehr Anschlüsse und mehr Makrozellen.

Wir merken uns:
Alle Makrozellen sind sich irgendwie ähnlich - aber nur auf den ersten Blick; in den Einzelheiten gibt es recht spitzfindige Unterschiede (es kann durchaus vorkommen, daß ein Schaltungsentwurf in keinen der industrieüblichen Schaltkreise paßt - weil bei jedem etwas anderes fehlt ...).

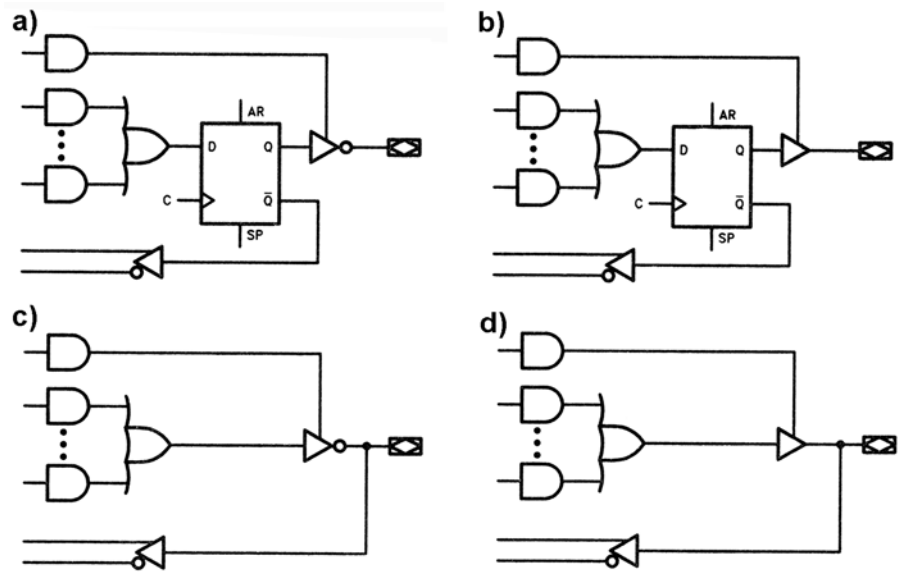


Abbildung 4.28 Konfigurationsbeispiele einer 22V10-Makrozelle

Erklärung:
a) - Register mit invertiertem Ausgang; b) - Register mit nichtinvertiertem Ausgang; c) kombinatorisch (Flipflop wird umgangen) mit invertiertem Ausgang; d) - kombinatorisch mit nichtinvertiertem Ausgang. Bei a) und b) ist der Flipflopausgang auf die UND-Anordnung zurückgeführt, bei c) und d) der E-A-Anschluß.

Kennwerte von GALs:
Durchlaufverzögerung: 1...25 ns, Taktfrequenz 20... > 100 MHz. Es gibt eine Vielzahl von Typen, teils mit besonderen Eigenschaften (besonders niedrige Speisespannung, besonders geringe Stromaufnahme, besonders schnell). Die maximale Taktfrequenz hängt wesentlich von der Nutzung ab (am schnellsten: ohne jegliche Rückführungen, am langsamsten: wenn Schaltkreisausgänge auf Schaltkreiseingänge zurückgeführt sind).

4.5. CPLDs

Komplexe programmierbare Logikschaltkreise haben - im Vergleich zu "gewöhnlichen" PALs und GALs - komplexere Makrozellen (mehr Funktionen, flexibler programmierbar) und vielseitiger nutzbare Schaltkreisanschlüsse. Manche Typen bieten zudem kombinatorische Verknüpfungsmöglichkeiten, die über einfache UND-ODER-Strukturen hinausgehen. Wir wollen den Aufbau derartiger Schaltkreise anhand von zwei Beispielen veranschaulichen (Abbildungen 4.29 bis 4.34).

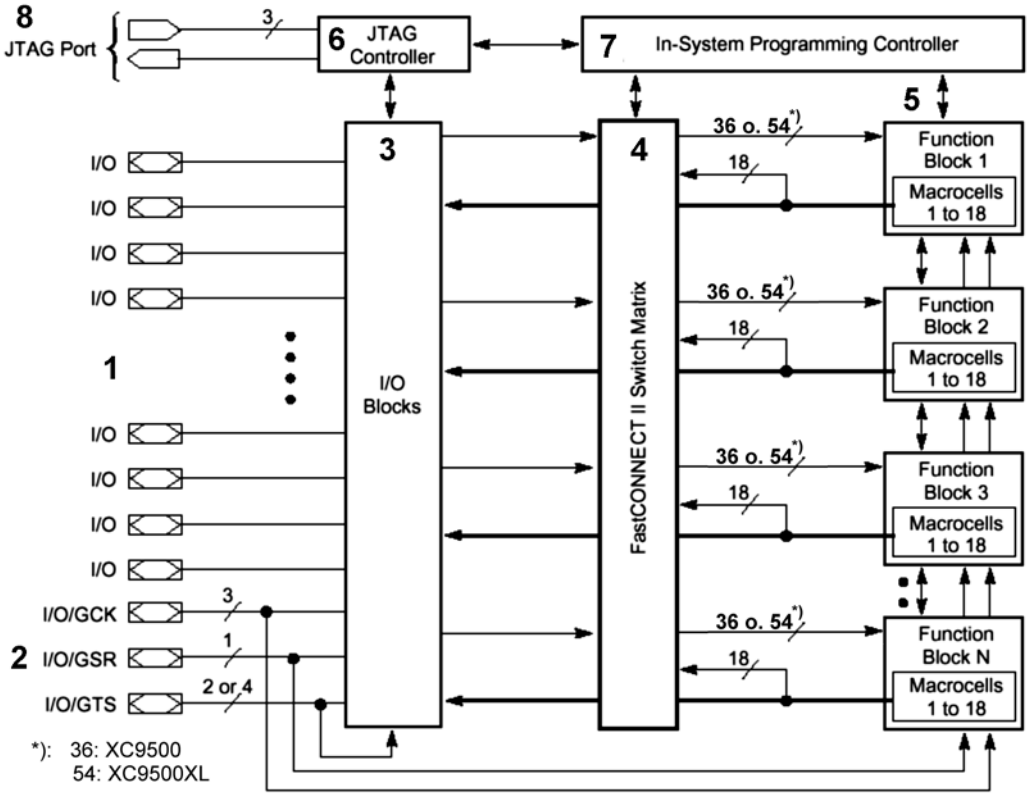


Abbildung 4.29 Eine CPLD-Schaltkreisfamilie im Blockschaltbild (XC9500/9500XL; Xilinx)

Erklärung:
1 - E-A-Anschlüsse; 2 - Takt- und Rücksetzanschlüsse; 3 - E-A-Blöcke; 4 - Koppelfeld; 5 - Funktionsblöcke mit Makrozellen; 6 - Steuerung der Programmierschnittstelle; 7 - Programmiersteuerung; 8 - JTAG-Programmierschnittstelle. Die Programmierung beruht auf Flash-Speicherzellen. Die CPLDs können in eingebautem Zustand programmiert werden (In-System Programming über die JTAG-Schnittstelle 8). Die verschiedenen Schaltkreise unterscheiden sich vor allem in der Anzahl der Funktionsblöcke und der E-A-Anschlüsse. Der einzelne Funktionsblock 5 ist eine Anordnung aus 18 Makrozellen mit vorgeschalteten UND-ODER-Verknüpfungen (Abbildung 4.30).

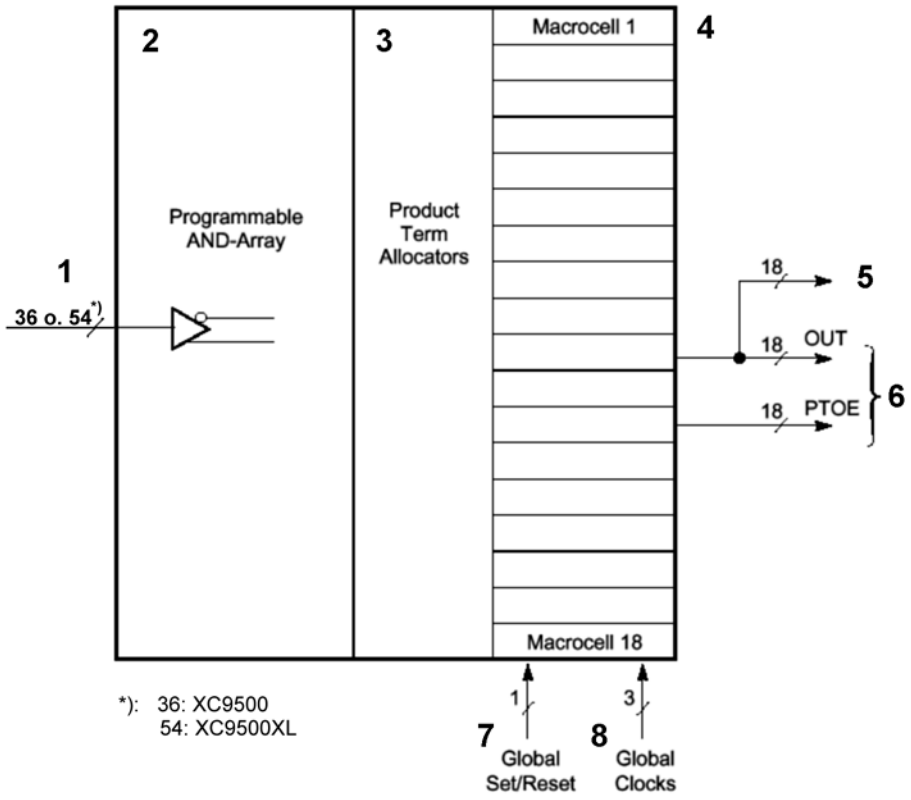


Abbildung 4.30 Ein Funktionsblock im Überblick (XC9500/9500XL; Xilinx)

Erklärung:
1 - 36 oder 54 Eingänge aus dem Koppelfeld; 2 - UND-Anordnung (90 Produktterme zu 36 oder 54 Variablen); 3 - Produkttermzuordner; 4 - Makrozellen (Abbildung 4.31); 5 - Ausgänge zum Koppelfeld; 6 - Ausgänge zu den E-A-Blöcken (OUT = Ausgangssignal, PTOE = Aufschalterlaubnissignal); 7 - globales Rücksetzsignal; 8 - globale Takte.

Globale Signale: Takt-, Setz- und Rücksetzsignale können global - alle Makrozellen des Schaltkreises betreffend - über bestimmte Anschlüsse (Position 2 in Abbildung 4.29) zugeführt werden. Wird ein globales Signal nicht benötigt, so ist der betreffende Anschluß als allgemeiner E-A-Anschluß verfügbar.

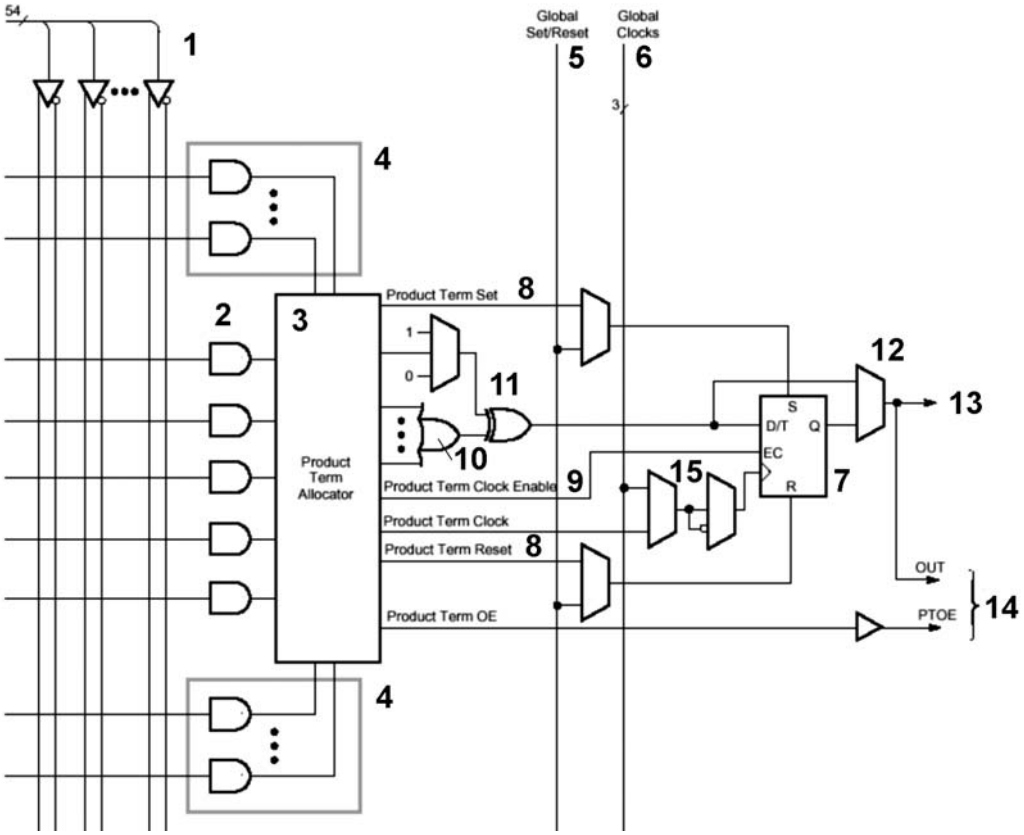


Abbildung 4.31 Eine Makrozelle im Blockschaltbild (XC9500XL; Xilinx)

Erklärung:
1 - Eingänge vom Koppelfeld; 2 - Produkttermbildung; 3 - Produkttermzuordner; 4 - weitere Produktterme; 5 - globales Setzen und Rücksetzen; 6 - globale Takte; 7 - Flipflop; 8 - Setzsignalauswahl; 9 - Rücksetzsignalauswahl; 10 - ODER-Verknüpfung; 11 - steuerbare Negation; 12 - Umgehung des Flipflops (Bypass); 13 - Ausgang zum Koppelfeld; 14 - Ausgänge zu den E-A-Blöcken; 15 - Taktauswahl (Quelle und Polarität).

Produktterme und Produkttermzuordner: jede Makrozelle bildet intern 5 Produktterme (UND-Verknüpfungen) über die 54 Signale (36 bei XC9500), die aus dem Koppelfeld kommen. Davon können einzelne Produktterme zum Bilden besonderer Signale verwendet werden (Setzen, Rücksetzen, Negation des Datensignals), Takt, Aufschalterlaubnis). Der Produkttermzuordner 3 schaltet die Produktterme entweder zu diesen Signalen durch oder führt sie auf die ODER-Verknüpfung 10. Im Extremfall enthält eine Makrozelle ein Fünffach-ODER über 5 108-fach-UNDs (54 Variable negiert und nichtnegiert). Die Besonderheit: die Makrozelle kann sich Produktterme von benachbarten Zellen gleichsam borgen (Position 4 in Abbildung 4.31 - die Spitzfindigkeiten wollen wir hier beiseite lassen ...)

Das Flipflop: es kann als D-Flipflop oder als T-Flipflop programmiert werden. Zudem hat es einen Takterlaubniseingang (Position 9 in Abbildung 4.31), kann also auch mit gesteuertem Takt bzw. im Sinne eines DV-Flipflops (vgl. Abschnitt 2.2.4.) betrieben werden.

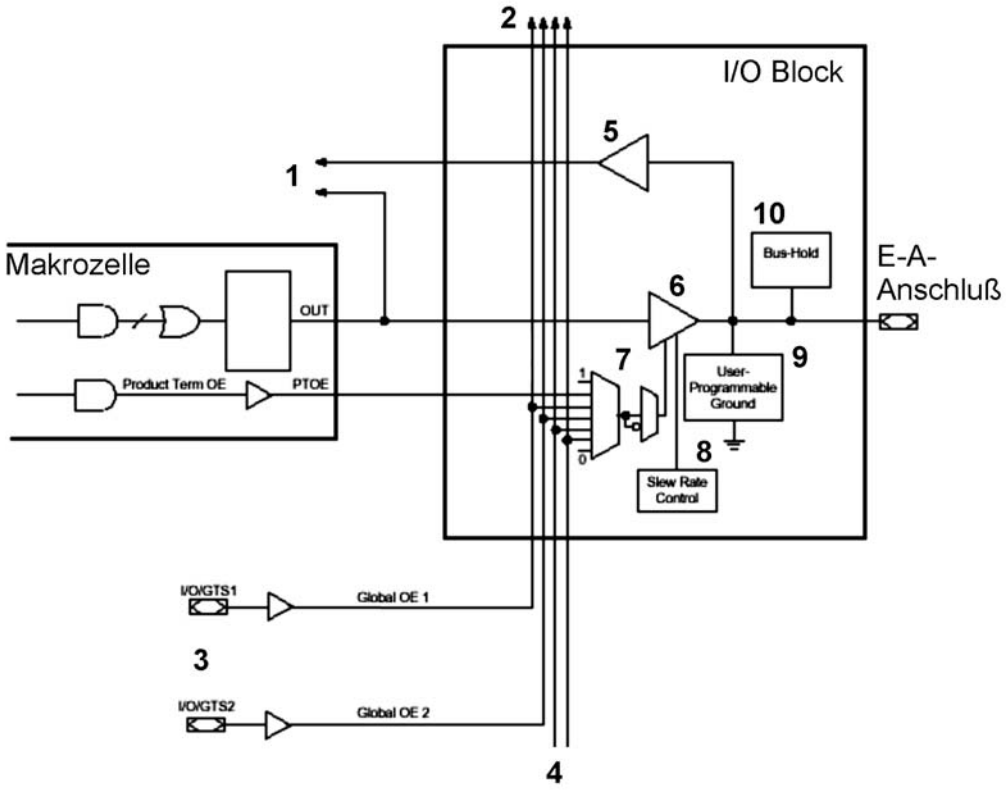


Abbildung 4.32 Ein E-A-Block im Blockschaltbild (Xilinx)

Erklärung:
1 - Eingänge zum Koppelfeld; 2, 3, 4 - globale Erlaubnissignale (zu allen E-A-Blöcken); 5 - Eingangspuffer; 6 - Ausgangstreiber; 7 - Aufschalterlaubnisauswahl (nie/immer/eines der globalen Signale/UND-Term aus Makrozelle); 8 - Programmierung der Flankensteilheit (schnell oder langsam); 9 - programmierbare Masseverbindung; 10 - Bushalteschaltung.

Jeder E-A-Anschluß ist auf einen E-A-Block geführt. Dort werden die Ein- und Ausgangswege voneinander getrennt. Die Eingänge werden zunächst auf das Koppelfeld geführt und können so beliebigen Funktionsblöcken und Makrozellen zugewiesen werden. Die Ausgangssignale sind hingegen an fest zugeordnete Makrozellen angeschlossen. Die Besonderheiten des E-A-Blocks:

- programmierbare Flankensteilheit. Sofern nicht höchste Geschwindigkeit gefordert ist, kann - im Interesse der Störsicherheit - die Flankensteilheit herabgesetzt werden.
- programmierbare Masseverbindung. Ein Digitalschaltkreis hat nie genug Masseverbindungen. Diese Vorkehrung ermöglicht es, ungenutzte Anschlüsse als zusätzliche Masseverbindungen auszunutzen.
- Bushalteschaltung. Hält einen inaktiven (hochohmigen) Anschluß auf dem jeweils letzten Pegel.

Das Koppelfeld (bei Xilinx: FastConnect Switch Matrix) ist jene Funktionseinheit, in der sich die CPLD-Architekturen vor allem unterscheiden. Xilinx hat hier ein echtes Crossbar-Netzwerk (Kreuzschienenverteiler) vorgesehen, das wirklich jeden seiner Eingänge zu jedem Funktionsblock durchschalten kann.

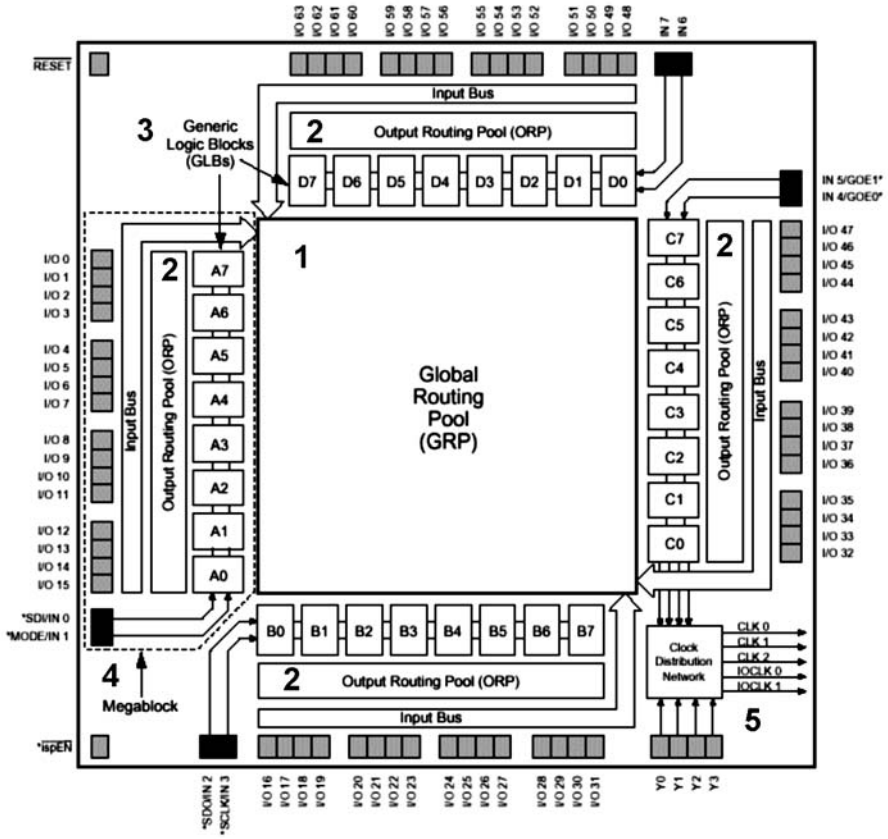


Abbildung 4.33 Eine weitere CPLD-Architektur (iSPLSI1000; Lattice)

Erklärung:
Eine Mischung aus Blockschaltbild und symbolischer Darstellung des Schaltkreis-Layouts. 1 - globales Koppelfeld; 2 - E-A-Koppelfelder; 3 - Funktionsblöcke; 4 - acht Funktionsblöcke sind zu einem sog. Megablock zusammengefaßt; 5 - Taktanschlüsse und Taktverteilung.

Die verschiedenen Hersteller haben verschiedene Begriffe. So heißt das Koppelfeld hier Routing Pool (und es gibt deren zwei - ein globales und in jedem Megablock eines für die Ausgangssignale). Die Funktionsblöcke heißen hier Generic Logic Blocks (GLBs; Abbildung 4.34).

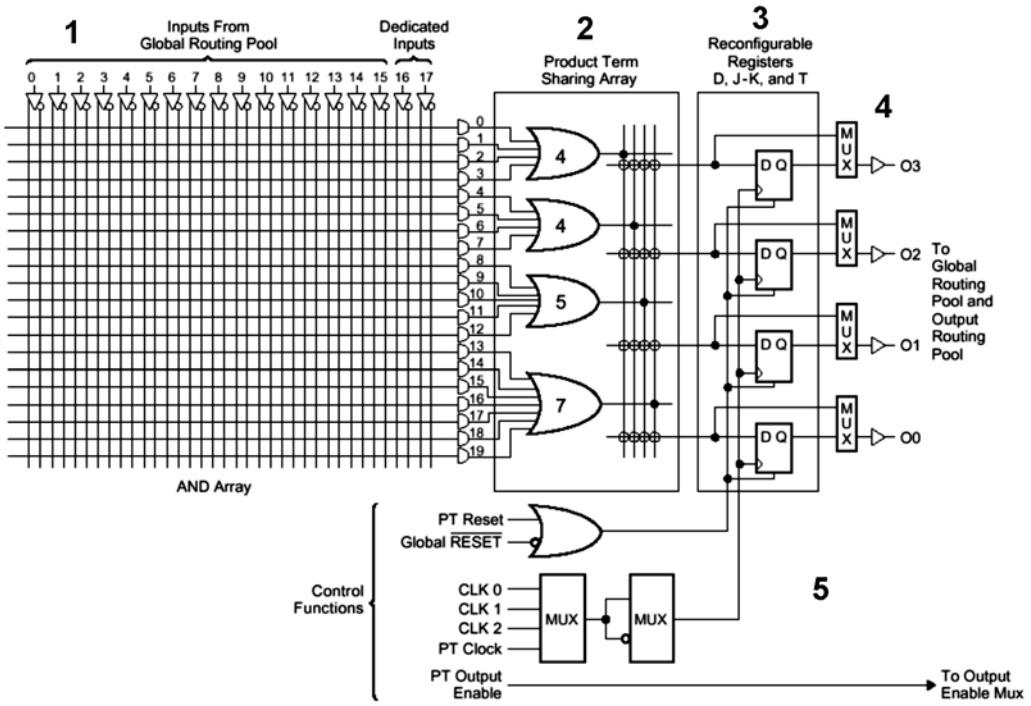


Abbildung 4.34 Ein Funktionsblock (Lattice)

Erklärung:
1 - UND-Anordnung; 2 - ODER-Verknüpfungen und Produkttermverteiler; 3 - Register mit programmierbarer Umgehung (Bypass); 4 - Ausgänge zu den Koppelfeldern; 5 - Rücksetzen, Takt, Aufschalterlaubnis.

Die UND-ODER-Struktur: jeder Funktionsblock (GLB) hat 18 Eingangssignale (16 aus dem globalen Koppelfeld und 2 lokale). Es werden insgesamt 20 Produktterme gebildet. Diesen sind 4 ODER-Verknüpfungen nachgeschaltet (2 vierfache, 1 fünffache und 1 siebenfache). Zum Rücksetzen, als Takt und als Aufschalterlaubnis wird je ein weiterer Produktterm gebildet. Die ODER-Verknüpfungen können über ein kleines Koppelfeld (hier: Produkttermverteiler (Product Term Sharing Array)) auf die Flipflops geschaltet werden. Die 4 Flipflops haben gemeinsame Takt- und Rücksetzeingänge. Sie können als D-, als T- und als JK-Flipflops programmiert werden.

Kennwerte von CPLDs
Durchlaufverzögerung (von Anschluß zu Anschluß): < 10... > 30 ns, Taktfrequenz 20... > 100 MHz. Es gibt eine Vielzahl von Typen, teils mit besonderen Eigenschaften (besonders niedrige Speisespannung, besonders geringe Stromaufnahme, besonders schnell). Anzahl der Makrozellen: von typischerweise 32...36 an aufwärts bis zu einigen hundert. Anzahl der Gatterfunktionen (eine Marketing-Angabe ohne allzu große Aussagekraft): von einigen hundert bis zu mehr als 10 000.

4.6. FPGAs (Field Programmable Gate Arrays)

FPGAs sind programmierbare Schaltkreise, mit denen die Hersteller Gebrauchseigenschaften in Hinsicht auf Integrationsgrad und Flexibilität bereitstellen wollen, die denen "echter" Gate Arrays nahekommen.

Das Prinzip: Man sieht eine programmierbare Verbindungsmatrix vor, in die Funktionsblöcke (als Zellen, Logik-Moduln, Logikblöcke o. ä. bezeichnet) eingebettet sind. Es gibt große Unterschiede zwischen den verschiedenen Architekturen (und viele Spitzfindigkeiten). Da wir nicht entwerfen wollen, begnügen wir uns mit einem kurzen Überblick über gängige Auslegungen (Abbildungen 4.35 bis 4.44).

Programmierung: die üblichen Programmierverfahren sind:

- Aufschmelzprinzip (Antifuse),
- Flash-ROM,
- RAM. Solche FPGAs müssen nach dem Einschalten erst einmal geladen werden. Das kann ein Prozessor oder Mikrocontroller erledigen. Die Schaltkreise haben aber typischerweise eine State Machine, die das Laden autonom steuern kann. Hierzu ist lediglich ein ROM oder EEPROM mit seriellem Interface anzuschließen.

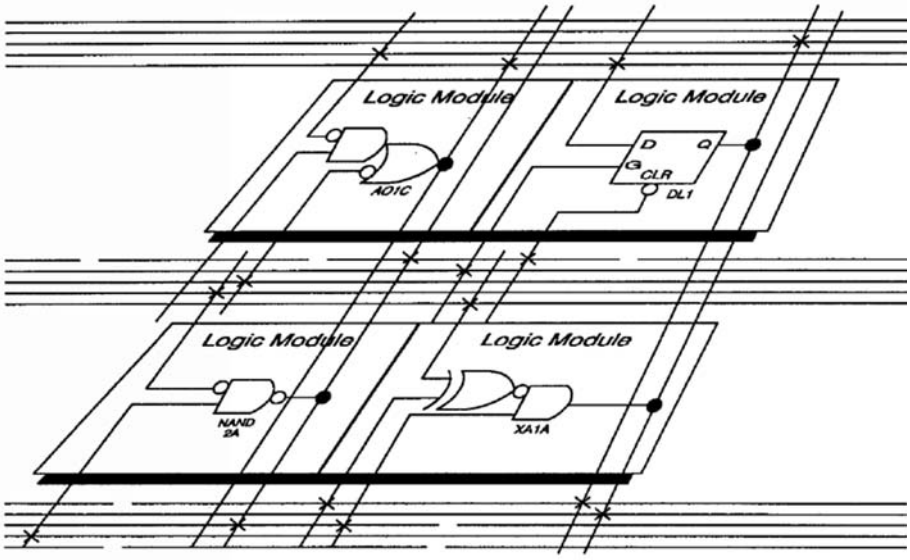


Abbildung 4.35 Der prinzipielle Aufbau eines FPGAs: Logik-Moduln zwischen Verbindungswegen (Actel)

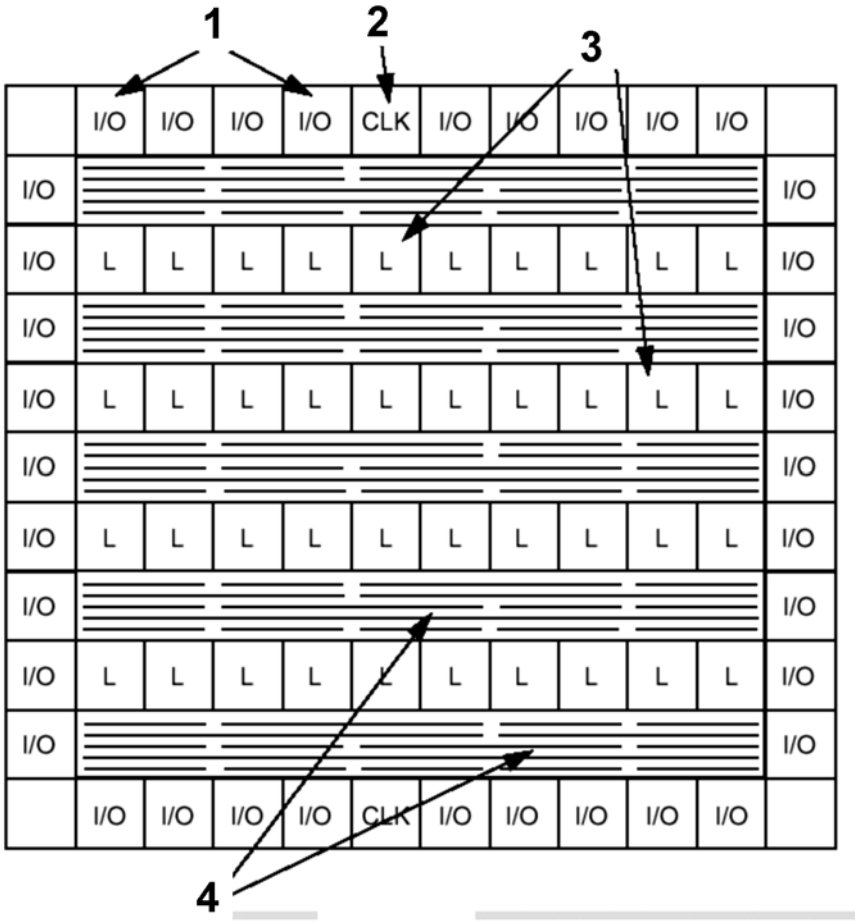


Abbildung 4.36 Eine FPGA-Schaltkreis im Überblick (Actel)

Erklärung:
1 - E-A-Blöcke; 2- Taktpuffer; 3 - Logikblöcke; 4 - Verbindungswege (Routing Tracks). Der Schaltkreis wird nach dem Antifuse-Prinzip programmiert.

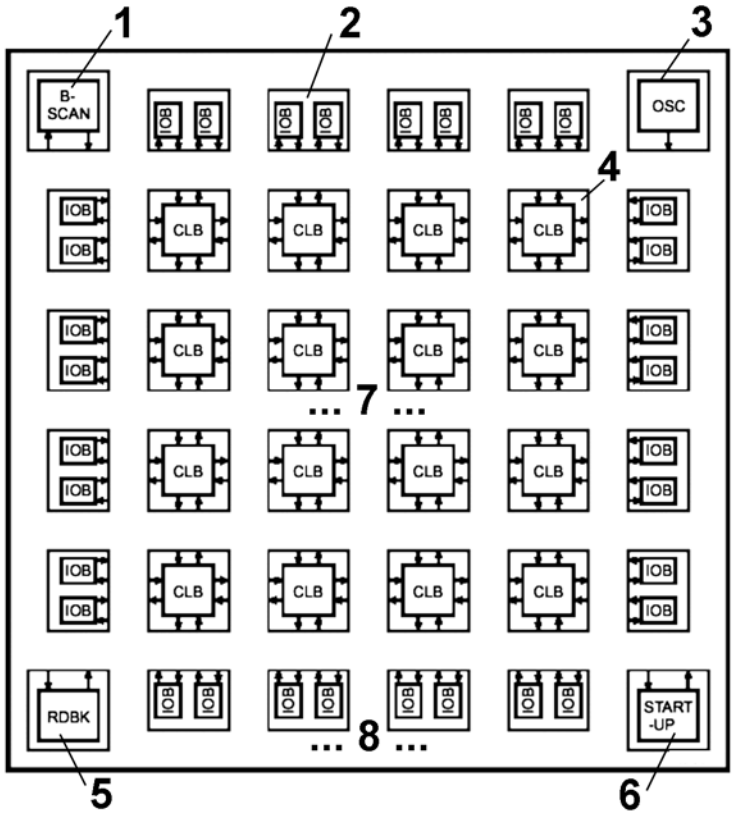


Abbildung 4.37 Ein weiterer FPGA-Schaltkreis (Xilinx)

Erklärung:
Der Schaltkreis wird nach dem RAM-Prinzip programmiert. 1 - Boundary-Scan-Schnittstelle (zur Anschluß- und Leiterplattenprüfung); 2 - E-A-Blöcke (IOBs); 3 - Taktgenerator; 4 - Logikblöcke (CLBs = Configurable Logic Blocks); 5 - Rücklesevorkehrungen (um die Programmierdaten zu Prüfzwecken zurücklesen zu können); 6 - Einschaltsteuerung und Ladeschnittstelle (z. B. zum Anschließen eines seriellen EEPROMs); 7 - CLB-Verdrahtungskanäle; 8- IOB-Verdrahtungskanäle (vgl. die Erklärung zu Abbildung 4.39).

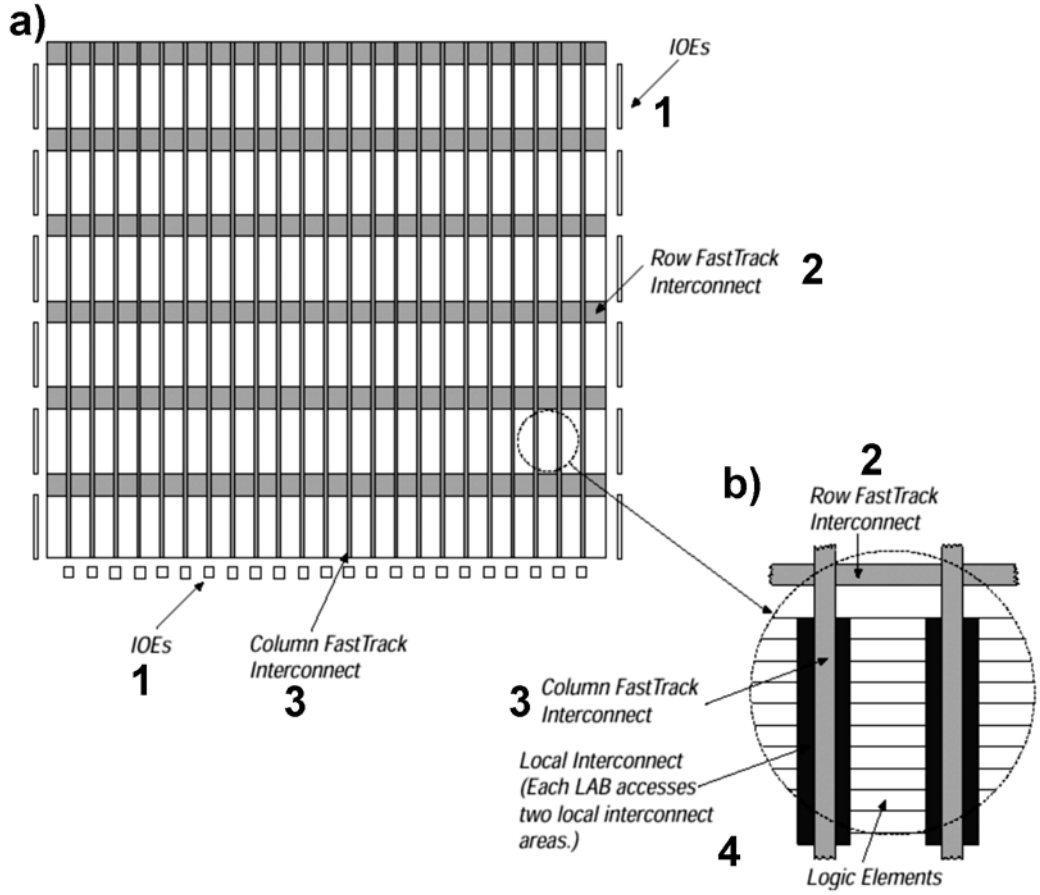


Abbildung 4.38 Globale und lokale Verdrahtung (Altera)

Erklärung:
Kleine FPGAs können ähnlich Abbildung 4.36 ausgelegt werden, nämlich mit einem System von Verbindungswegen, die sich in zeilen- oder spaltenförmiger Anordnung jeweils über den gesamten Schaltkreis ertrecken. Bei FPGAs mit vielen Logikblöcken funktioniert das nicht mehr: Leitungen in großen Schaltkreisen werden naturgemäß besonders lang - das verlängert die Schaltzeiten und erhöht die Störempfindlichkeit. Zudem sind es sehr viele Leitungen (= Siliziumfläche). Die Hersteller haben sich deshalb Konfigurationen aus Leitungssystemen verschiedener Länge einfallen lassen (und teils mit mehr oder weniger klangvollen Handelsnamen bezeichnet). In unserem Beispiel führen gitterförmig angeordnete Wege (die hier FastTrack Interconnect heißen) über den ganzen Schaltkreis (globale Signalwege). Dazwischen sind die Logikblöcke eingebettet, die von zusätzlichen lokalen Verbindungswegen umgeben sind. 1 - E-A-Blöcke; 2 - globale Zeilenleitungen; 3 - globale Spaltenleitungen; 4 - lokale Verbindungen zwischen benachbarten Logikblöcken (hier: Logic Elements).

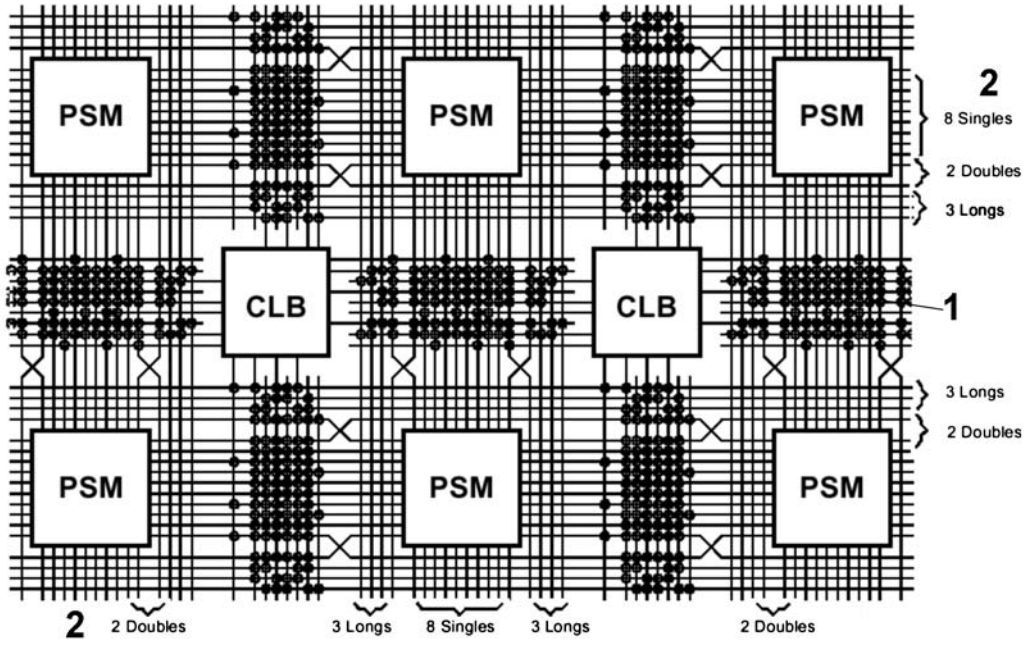


Abbildung 4.39 Verschieden lange Verbindungen (Xilinx)

Erklärung:
In dieser FPGA-Architektur unterscheidet man 3 Arten von Verbindungswegen:

- CLB Routing Channels, die in Zeilen und Spalten zwischen den Logikblöcken (hier: CLBs) geführt sind,
- IOB Routing Channels, die gleichsam als Ring um den Schaltkreisrand herumgeführt sind und die E-A-Blöcke mit den CLB Routing Channels verbinden,
- Global Routing Channels, die vor allem Taktsignale über den Schaltkreis verteilen.

Die Abbildung zeigt Logikblöcke (CLBs) und programmierbare Koppelfelder (PSMs), die zwischen CLB Routing Channels eingebettet sind. 1 - programmierbare Anschaltungen der CLBs; 2 - die Verbindungswege gibt es in 3 Längen: kurz, doppelt und lang (Single, Double, Long). An den Kreuzungen von Zeilen und Spalten sind programmierbare Koppelfelder angeordnet (PSM = Programmable Switch Matrix), die jede Zeilenleitung mit jeder Spaltenleitung verbinden können.

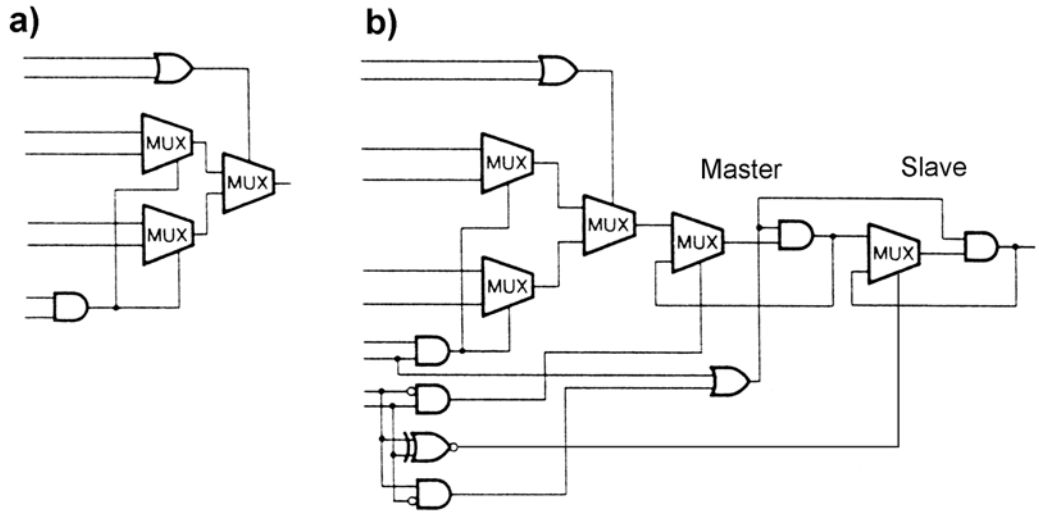


Abbildung 4.40 Einfache Logikblöcke (Texas Instruments/Actel)

Erklärung:

a) kombinatorischer Logikblock. Die Multiplexeranordung ermöglicht es, eine Vielzahl elementarer kombinatorischer Verknüpfungen zu realisieren, indem die Eingänge durch Programmierung mit Signalleitungen oder mit Festwerten (Low/High) verbunden werden.

b) sequentieller Logikblock. Einem kombinatorischen Logikblock ist eine Flipflopanordnung nachgeschaltet, die aus zwei Latches in Master-Slave-Anordnung besteht. Die Latches werden über vorgeordnete Gatter gesteuert. Indem man deren Eingänge entsprechend beschaltet (durch Programmieren), kann man die Anordnung auf eine von 3 möglichen Funktionen einstellen: (1) flankengesteuertes D-Flipflop, (2) D-Latch, (3) kombinatorische Durchreiche (vgl. auch S. 77 bis 79).

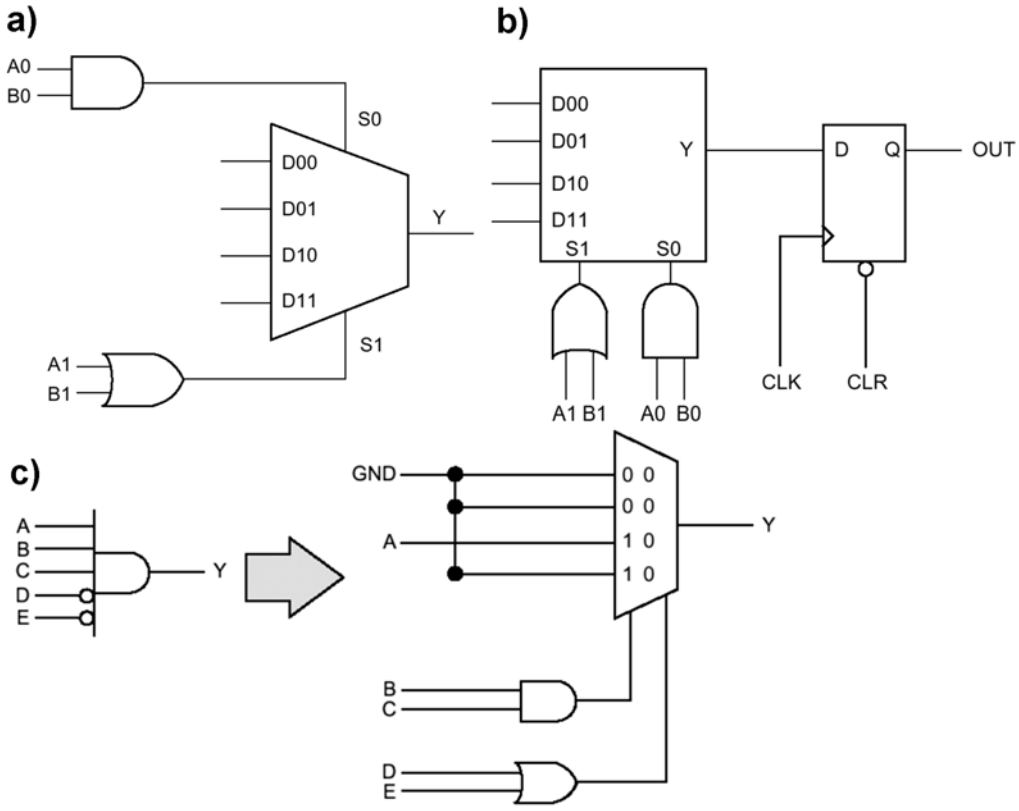


Abbildung 4.41 Erweiterte - aber immer noch überschaubare - Logikblöcke (Actel)

Erklärung:

a) kombinatorischer Logikblock, bestehend aus einem 4-zu-1-Multiplexer mit Auswahlsignalen, die UND-Gattern nachgeschaltet sind,

b) sequentieller Logikblock, bestehend aus einem kombinatorischen Logikblock mit nachgeschaltetem Speicherglied (Programmiervarianten: D-Flipflop, Latch, Umgehung),

c) so wird der Logikblock a) verschaltet, um die Fünffach-UND-Verknüpfung (links) zu realisieren (zum Prinzip vgl. Abschnitt 4.1.2.).

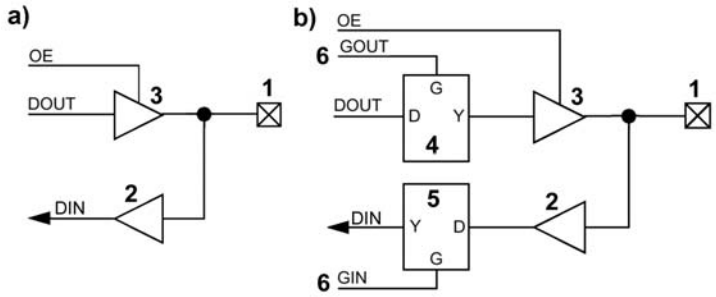


Abbildung 4.42 Zwei einfache E-A-Blöcke (Actel)

Erklärung zu Abbildung 4.42:
1 - E-A-Anschluß; 2 - Eingangspuffer; 3 - Tri-State-Treiber; 4 - Ausgangsspeicher; 5 - Eingangsspeicher; 6 - Taktsignale (Gate/Clock).

a) einfachste E-A-Anschaltung über Eingangspuffer und Tri-State-Treiber,

b) E-A-Block ähnlich a), aber mit ein- und ausgangsseitigen Speicherstufen. In manchen FPGAs handelt es sich um Latches, in manchen ist die Funktionsweise (Latch oder Flipflop) programmmierbar.

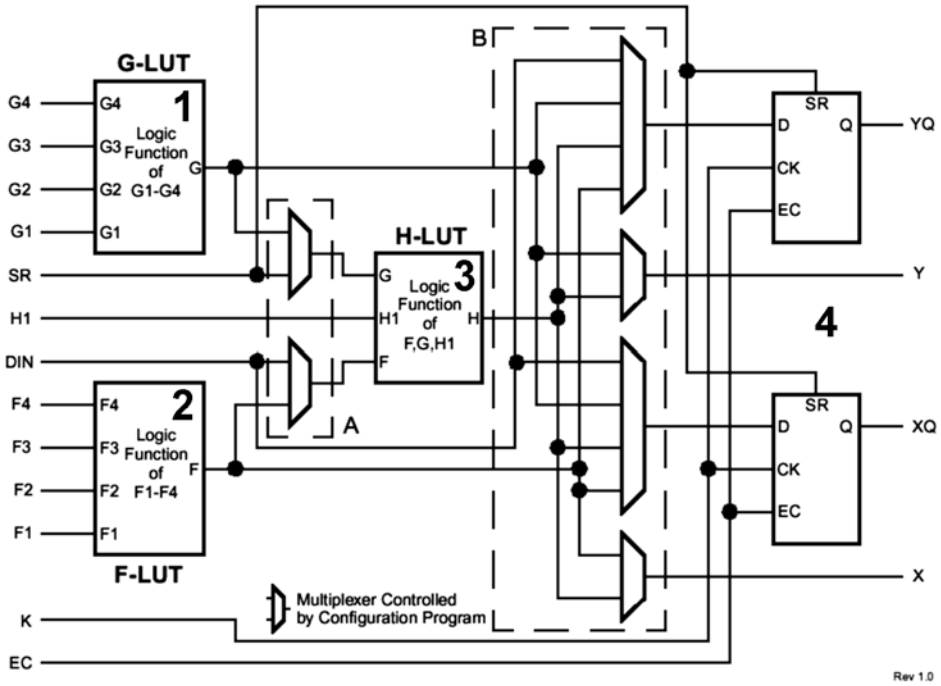


Abbildung 4.43 Ein höher entwickelter Logikblock (Xilinx)

Erklärung:
Dieser Logikblock (CLB; vgl. die Abbildungen 4.37 und 4.39) wird nach dem RAM-Prinzip programmiert. 1, 2 - 16-Bit-RAM; 3 - 8-Bit-RAM; 4 - Flipflopanordnung. Die RAMs wirken als Logikzuordner (hier: LUT = Lookup Table). Die RAMs 1 und 2 können jede kombinatorische Verknüpfung von 4 Variablen realisieren, RAM 3 eine Verknüpfung von 3 Variablen. Die dargestellten Multiplexer werden durch Programmierung gestellt (es sind keine Schaltmittel, über die der Anwender frei verfügen kann).

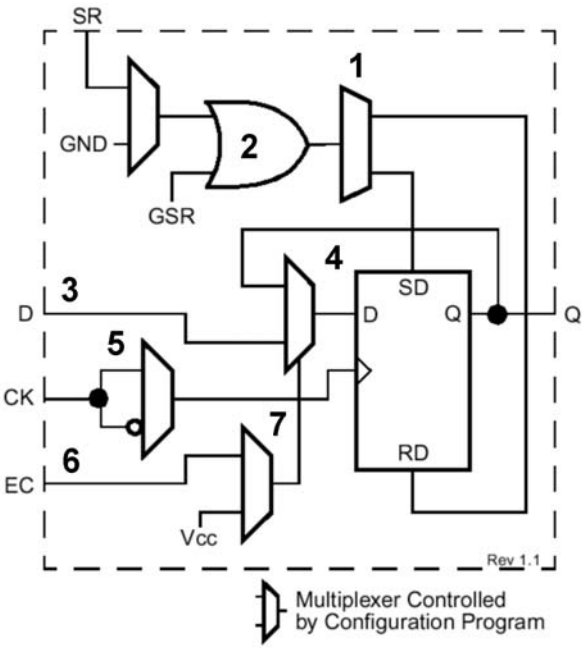


Abbildung 4.44 Einer der Flipflops von Abbildung 4.43 im einzelnen (Xilinx)

Erklärung:
1 - Auswahl zwischen Setzen und Rücksetzen; 2 - Invertierung des Setz- oder Rücksetzsignals; 3 - Dateneingang (vom 3. RAM); 4 - Rückführung (Selbsthaltung); 5 - Polaritätswahl für Taktsignal; 6 - Erlaubnissignal; 7 - Auswahl zwischen D- und DV-Verhalten (D: Dateneingang 3 zum Flipflop durchgeschaltet, keine Selbsthaltung; DV - Selbsthaltung ist wirksam, solange Erlaubnissignal (EC) inaktiv - vgl. Abbildung 2.13). Die dargestellten Multiplexer werden durch Programmierung gestellt (es ist nicht möglich, während des Betriebs z. B. das Verhalten des Flipflops zwischen D und DV umzuschalten).

Kennwerte von FPGAs
Unterscheiden sich die einzelnen CPLD-Familien schon beachtlich voneinander (vgl. die Abbildungen 4.29 und 4.33), so sind die verschiedenen FPGA-Architekturen kaum noch miteinander vergleichbar. Zudem gibt es FPGAs für verschiedene Speisespannungen und mit (teils programmierbaren) Schnittstellen für verschiedene Logikspezifikationen (LVTTL, LCVCMOS, GTL, SSTL, AGP, PCI usw.). Die Anzahl der “brauchbaren Gatter” (Usable Gates) ist eine gängige Marketing-Angabe. Werte: von einigen hundert bis zu einigen Millionen. Was aber leistet so ein Gatter? Wievielen solcher Gatter entspricht eine einzelner Logikblock? - Dazu hält man sich typischerweise sehr bedeckt ... Auch Angaben zu Verzögerungszeiten und Taktfrequenzen können nur sehr rohe Richtwerte sein (tatsächlich hängt die jeweilige maximale Taktfrequenz vom Schaltungsentwurf ab - und davon, was die Entwurfssoftware daraus gemacht hat. Beispielhafte Angaben zu modernen Typen: Schaltzeiten von Taktflanke zum Ausgangsanschluß: 2...5 ns, Verzögerung innerhalb eines Logikblocks: > 1... < 0,3 ns, Verzögerung von Anschluß zu Anschluß: 10 ns ... 2 ns; maximale Taktfrequenzen: 25... > 133 MHz.

Vor allem die besonders dicken FPGAs sind keine Billigschaltkreise - sie sind viel aufwendiger

als in vergleichbarer Technologie gefertigte Hochleistungsprozessoren. Beispiel: die Virtex-Baureihe von Xilinx. Die Gatterangabe (hier. “System Gates” reicht von 40 000 an aufwärts bis zu 10 Millionen. Ein Schaltkreis mit 4 Millonen solcher “System Gates” hat 5760 CLBs zu je 4 Anordnungen ähnlich Abbildung 4.43 (die aber etwas komplizierter sind) und 912 E-A-Anschlüsse (Anzahl der Flipflops: 46 080). Der Aufwand zur Programmierung (hier: nach dem RAM-Prinzip) ist gewaltig: die Schaltkreise enthalten mehr als 100 Millionen Transistoren - es würden mehr als 12 P6- oder mehr als 3 Pentium-4-Prozessoren draufpassen (zudem könnte man die Prozessoren intern mit Taktfrequenzen im GHz-Bereich betreiben, während für die FPGAs an den Anschlüssen zwischen 100 und 200 MHz und im Innern ca. 500 MHz spezifiziert werden). Solche Schaltkreise kommen also wirklich nur für Sonderanwendungen in Frage, wo mit Prozessoren nicht viel auszurichten ist und wo das Geld keine so große Rolle spielt (Verschlüsselung (wohl auch: Knacken von Schlüsseln ...), Bilderkennung, Telekommunikationseinrichtungen am "oberen Ende" der Vermittlungshierarchie usw.).