Heft 05
Programmierbare Logik

Stand: 1.02

- Vorläufiges Exemplar. Nur zur Information -

1. Grundlagen

1.1. Weshalb universelle und 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 1.1).

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

Tabelle 1.1 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: (1) die Festlegung der endgültigen Funktion des Schaltkreises darf keine Halbleiter-Fertigungsschritte mehr erfordern, (2) der technologisch mögliche Integrationsgrad muß möglichst wirksam genutzt werden können.

Vergegenwärtigen wir uns ein Problem der Elementarschaltungen: Wenn man Freiheit (hinsichtlich der Nutzung) haben will, müssen alle Ein- und Ausgänge der Gatter, Flipflops, Register, Zähler usw. *zugänglich* sein. Das bedeutet praktisch, daß sie - bei herkömmlicher Verbindungstechnologie (Leiterplatte, Draht) - aus dem Schaltkreis *herausgeführt* werden müssen. Anders herum: Gehäusegröße und Anschlußzahl begrenzen die praktisch nutzbare Funktionalität (an sich könnte man auf dem Silizium viel mehr Funktionen unterbringen).

Zudem ergeben sich Einbußen an Geschwindigkeit und Zuverlässigkeit.

Der Schaltkreishersteller, der einschlägige Lösungen anbieten will, steht somit vor der Aufgabe, von Hause aus *universelle* Schaltkreise zu entwickeln, die man - auf möglichst bequeme Weise - "zurichten" kann, um weitgehend beliebige anwendungsseitige Funktionen auszuführen.

1.2. Die schaltalgebraischen Grundlagen

1.3.2. Universelle kombinatorische Schaltungen

Die entscheidende Grundlage bildet die Darstellung der zu realisierenden Schaltfunktion als Wahrheitstabelle oder als disjunktive Normalform DNF (in anderer Bezeichnungsweise: Sum-of-Products-Darstellung; SOP). Kanonische disjunktive Normalform (KDNF) und Wahrheitstabelle entsprechen unmittelbar einander. Jede Zeile der Wahrheitstabelle, in der die Schaltfunktion als "erfüllt" gekennzeichnet ist, steht für einen Konjunktions- bzw. Produktterm. Alle Produktterme, die die Schaltfunktion erfüllen, werden schließlich disjunktiv verknüpft. Auf dieser Grundlage bieten sich zwei Ansätze, um universelle kombinatorische Schaltungen zu bauen:

1. wir speichern die gesamte Wahrheitstabelle und sehen für jede gegebene Belegung nach, ob diese die Schaltfunktion erfüllt oder nicht.
2. wir bauen eine universelle KDNF-Hardware. Wenn wir jede beliebige Verknüpfung zwischen n Variablen verwirklichen wollen, müssen wir mit allen Möglichkeiten rechnen. Die KDNF hat 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 1.1). 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 1.1 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 1.1 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 1.2).

Abbildung 1.2 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 Auswahlnetzwerk und Decoder für die Auswahladresse. Ein Multiplexer mit n Adreßeingängen erlaubt es, einen von 2^n Daten-Eingängen auf den Ausgang durchzuschalten. Wir können somit einen 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 1.3).

Abbildung 1.3 Verwirklichung einer Schaltfunktion mittels Multiplexer (1)

Man kann sogar die Variablen-Anzahl noch um 1 erhöhen. Mit anderen Worten: ein Multiplexer mit n Adreßeingängen kann beliebige Schaltfunktionen über n+1 Variable verwirklichen. Üblicherweise kennzeichnet man einen Multiplexer nicht durch die Anzahl seiner Adreß- sondern durch die seiner Dateneingänge. Damit lautet der Satz: ein 1-aus-n-Multiplexer ermöglicht es, eine beliebige Schaltfunktion über (ld (n)) +1 Variable zu realisieren (also beispielsweise ein 1-aus-8-Multiplexer eine Funktion über 4 Variable). Voraussetzung dafür ist, daß jeder Dateneingang gemäß der Wahrheitstabelle mit jeweils einem von vier Werten belegt wird: (1) mit dem Festwert "0", (2) mit dem Festwert "1", (3) mit einer der Eingangsvariablen direkt, (4) mit der besagten Eingangsvariablen negiert (Abbildung 1.4).

Abbildung 1.4 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 1.1. 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 1.5). In anderer Sichtweise: ein adressierbarer Speicher kann die gesamte Wahrheitstabelle aufnehmen (eine solche gespeicherte Tabelle bezeichnet man im Englischen als Lookup Table LUT).

Abbildung 1.5 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.

Aufwandsersparnis

2^n UND-Gatter bedeuten, namentlich wenn n größer wird, einen beachtlichen Aufwand, auch wenn es Lösungen gibt, ihn in gewissen Grenzen erträglich zu halten (Stichwort: Matrix-Organisation). 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 beschaltet werden (Abbildung 1.6).

Abbildung 1.6 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 Aufrufbreite; für "echte" UND-ODER-Strukturen

(Schaltungen nach Abbildung 1.6) gibt es zwei Auslegungsmöglichkeiten (Abbildung 1.7):

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 1.7 Realisierung mehrerer Schaltfunktionen

1.3.3. 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 1.8 zeigt ein "klassisches" Beispiel.

Abbildung 1.8 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.)

In der Praxis verwendet man (3 Eingänge sind meist zuwenig) Multiplexer-Gatter-Kombinationen oder kleine Speicheranordnungen (Lookup Tables). In folgenden Abschnitten werden wir einige Beispiele kennenlernen.

1.3.4. 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 ausgangsseitig angeordnet. Sie sind entweder als einfache Register oder als - ihrerseits programmierbare - "Makrozellen" (Output Logic Macrocells; OLMCs) organisiert (Abbildung1.9).

Abbildung 1.9 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, Vorder- oder Rückflankensteuerung),
- die Polarität (direkt oder negierend) und den Zustand des zugehörigen Schaltkreisanschlusses einstellen,
- interne Rückführungen auf die Kombinatorik schalten,
- ergänzend zur vorgeordneten UND-ODER-Struktur die Eingänge des Flipflops weiteren Signalwandlungen unterziehen (einfachstes Beispiel: Negation).

Einige Beispiele werden wir in den folgenden Abschnitten kennenlernen.

1.3.5. Programmieren

Das "Zurichten" der universellen Hardware
Kombinatorische Netzwerke auf Grundlage fertiger (off the shelf) Decoder- oder Multiplexerschaltkreise werden durch die Verdrahtung (bzw. Leiterzugführung) gebildet, also *außerhalb* des Schaltkreises. Sind Einstellungen *innerhalb* von Schaltkreisen notwendig, so ist ein "Verdrahten" naturgemäß unmöglich. Vielmehr müssen elektrische Funktionseigenschaften oder physikalische Effekte genutzt werden, um die jeweiligen Einstellungen einzubringen und zu halten. Man spricht hierbei vom *Programmieren.*

Programmierverfahren
Tabelle 1.2 gibt eine Übersicht.

Programmierverfahren	Ausführung	Änderbarkeit	Bemerkungen
Maskenprogrammierung	beim Halbleiter-hersteller	im einzelnen Schaltkreis nicht mehr änderbar	nur in ROMs von praktischer Bedeutung
Durchschmelzprinzip (Fuse)	beim Anwender	im einzelnen Schaltkreis praktisch nicht mehr änderbar	alle Verbindungen sind vorgefertigt, die nicht benötigten werden beim Programmieren getrennt
Aufschmelzprinzip (Antifuse)	beim Anwender	im einzelnen Schaltkreis praktisch nicht mehr änderbar	alle Verbindungsstellen sind zunächst getrennt, benötigte Verbindungen werden beim Programmieren hergestellt
Ladungsspeicherung mit UV-Löschung	beim Anwender	durch Löschen und Neuprogrammieren	Löschen durch UV-Licht; erfordert Quarzglasfenster im Schaltkreis (es gibt auch preisgünstige Ausführungen ohne Fenster; diese kann man nicht mehr löschen)
Ladungsspeicherung mit elektrischer Löschung	beim Anwender	durch Löschen und Neuprogrammieren	Löschen durch elektrische Impulse
RAM-Zellen	beim Anwender bzw. während des Betriebs	durch Umladen; auch während des normalen Betriebs beliebig oft möglich	Halten der Information in Flipflops bzw. Latches; nach jedem Einschalten ist erneutes Laden erforderlich

Tabelle 1.2 Programmierverfahren: eine Übersicht

Rückleseschutz

Die programmierten Einstellungen, Verbindungen usw. lassen sich (bei den meisten Schaltkreistypen) im Programmierbetrieb auch wieder auslesen. Für Testzwecke (um nachzusehen, ob das Programmieren auch geklappt hat) ist dies eine sehr nützliche Vorkehrung. Sie ermöglicht aber auch, den "Inhalt" eines programmierten Schaltkreises einfach zu kopieren. Viele Schaltkreise haben deshalb einen programmierbaren Rücklese- bzw. Kopierschutz (Security Fuse; Security Cell). Die Wirkung: Nach dem Überprüfen des programmierten Inhalts (durch Zurücklesen) wird eine zusätzliche Bitposition programmiert, über die dann das weitere Zurücklesen verhindert wird.

Hat so etwas überhaupt Sinn? - Jeder, der den Schaltkreis in der Hand hat und den Entwurf stehlen will, kann doch einfach alle möglichen Belegungen durchprobieren. Bei reiner Kombinatorik und eher geringen Eingangszahlen (um 20) ist dies tatsächlich kaum ein Problem. (Beispiel: einTestautomat braucht 1 µs je Kombination. 2^{20} Kombinationen wären dann in 1 s abgeprüft.) Aber bei sequentiellen Schaltungen... (Wir wollen hier nicht weiter in Einzelheiten gehen. Die Automatentheorie zeigt auf, daß man schon bei recht einfachen sequentiellen Schaltungen geradezu astronomische Zeiten braucht, um deren Verhalten allein durch Herumprobieren an den Eingängen und Beobachten der Ausgänge zu bestimmen.)

Ist der Schaltkreis löschbar, wird die Security Cell bei jedem Löschen mitgelöscht.

Elektronische Signatur

In vielen programmierbaren Schaltkreisen ist eine Hersteller- und Typkennung gespeichert, die im Programmierbetrieb ausgelesen werden kann (Electronic Signature). Auch manche Speicherschaltkreise haben solche Vorkehrungen (im besonderen EPROMs). So erlauben übliche GALs, 64 Bits zu speichern, die frei verwendet werden können, um beispielsweise eine Kennung des Herstellers (der den Schaltkreis programmiert hat), eine Teilnummer, ein Fertigungsdatum usw. zu speichern. Diese Angaben können jederzeit wieder gelesen werden (auch bei aktiviertem Rückleseschutz).

Was ist zu programmieren?

Betrachten wir wieder unsere kombinatorische Hardware: 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 aber 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 1.10).

Abbildung 1.10 Kurzdarstellung programmierbarer Verbindungen

1.3.6. Programmierbare Logikschaltkreise: eine Übersicht

Elementare kombinatorische Schaltkreise werden als UND-ODER-Strukturen aufgefaßt und danach unterschieden, welche Schaltungsteile von vornherein festgelegt und welche durch Programmierung veränderbar sind (Abbildung 1.11; vgl. auch Abbildung 1.7):

1. UND fest (zudem vollständig, also 2^n UNDs bei n Eingängen), ODER programmierbar: adressierbarer **Speicher** (in den meisten Fällen werden PROMs eingesetzt),
2. UND-Eingänge programmierbar, ODER programmierbar: **PLA** (Programmable Logic Array),
3. UND-Eingänge programmierbar, ODER-Eingänge fest zugeordnet: **PAL** (Programmable Array Logic).

Abbildung 1.11 Elementare programmierbare Logikstrukturen im Vergleich (National Semiconductor)

Begriffsbildungen
Die Begriffe werden teils herstellerspezifisch verwendet, teils hat sich ein allgemeiner Gebrauch durchgesetzt (obwohl es sich manchmal im eigentlichen Sinne um Warenzeichen handelt).

Programmable Logic Devices (PLDs)
Dies ist der übliche Oberbegriff für programmierbare Logikschaltkreise. **EPLD** bezeichnet üblicherweise eine elektrisch (mehrmals) programmierbare und durch UV-Licht löschbare Schaltung.

Programmable Logic Arrays (PLAs)
Ein PLA (vgl. Abbildung 1.11) ist praktisch eine 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 (vgl. Abbildung 1.11). PAL-Schaltkreise werden zumeist mittels Durchschmelzverfahren (Fusible Link) programmiert.

Generic Array Logic (GALs)
Ein GAL-Schaltkreis ist eine elektrisch programmier- und löschbare, um Makrozellen (vgl. Abbildung 1.9) 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.)

Anwenderseitig programmierbare Schaltungskomplexe (Field Programmable Gate Arrays; FPGAs)

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

Registerd PROMs, State Machines und Mikrocontroller
Derartige Schaltkreise gehören zur "programmierbaren Logik", obwohl ihr Anwendungsgebiet in gewissem Sinne eingeschränkt ist (nämlich auf die Realisierung von Funktionen, die sich vorteilhaft als Zustandsautomat (State Machine) verwirklichen lassen).

Registered und State Machine PROMs können in jedem Taktzyklus einen Zustandswechsel ausführen. Wenn zwischen den Zustandswechseln genügend Zeit bleibt, kann man auch Mikrocontroller bzw. -prozessoren, entsprechend programmiert, als State Machines verwenden. Im besonderen eignen sich hierfür Typen, die durch ihre Architektur die Gewähr bieten, die Zustandswechselzeiten stets einhalten zu können (Stichwort: Vorhersagbarkeit des Realzeitverhaltens).

Abbildung 1.12 gibt einen Überblick über die verschiedenen Arten programmierbarer Schaltkreise und nennt einige Hersteller.

Abbildung 1.12 Programmierbare Schaltkreise: eine Übersicht (Texas Instruments)

Entwicklungsmethodik
Es ist praktisch unmöglich, einen PLD-Schaltkreis "von Hand" bis aufs Bit zu programmieren. Elementare Entwurfsunterstützungs-Software gestattet es, Boolesche Gleichungen oder Wahrheitstabellen einzugeben. Komplexe Systeme, wie man sie braucht, um beispielsweise FPGAs sinnvoll nutzen zu können, gestatten es, Schaltpläne herkömmlicher Art zu erfassen. Das heißt, der Entwickler arbeitet weiterhin mit Gattern, Decodern, Multiplexern, Registern, Zählern usw., wie er es bisher gewohnt war. (Derartige Funktionselemente, die im Rahmen der Entwicklungssoftware zur Verfügung gestellt werden, heißen "Softmakros".) Die Software leitet dann aus der eingegebenen Schaltung die Programmierung der Funktionsblöcke und der Verbindungen zwischen ihnen ab. Auf

Funktionsschaltplänen, die das "Innere" beispielsweise eines FPGA dokumentieren, finden Sie deshalb die gewohnten Symbole und keine Einzeldarstellungen von Funktionsblöcken oder Verbindungsmatrizen.

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 werden nach dem Durchschmelzprinzip (Fusible Link) programmiert.

Wichtige Kennwerte

Die genaue Dokumentation programmierbarer Schaltkreise erfordert umfangreiche Datenbücher. Der funktionellen Komplexität entsprechen naturgemäß vielfältige statische und dynamische Kennwerte (Parameter). Wenn Sie es genau wissen wollen, bleibt Ihnen der Blick ins Datenmaterial 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 ein sequentieller Schaltkreis 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).

Bei sehr komplexen Schaltungen (FPGAs) kommt noch die Anzahl der nutzbaren Gatter (Usable Gate Count) hinzu.

Hinweise:

1. Unsere Angaben in den folgenden Abschnitten haben rein näherungsweisen, überblicksmäßigen Charakter.
2. Viele Schaltkreistypen werden in unterschiedlich "schnellen" Versionen gefertigt.
3. Je komplexer der Schaltkreis, um so mehr hängen Verzögerungszeiten und Taktfrequenzen von der tatsächlich verwirklichten Schaltungsstruktur ab, d. h. davon, wie die einzelnen Funktionsblöcke in sich programmiert und untereinander verbunden sind.

Bezeichnung elementarer programmierbarer Schaltkreise

Für PALs und GALs ist ein Bezeichnungssystem üblich, das einen groben Überblick über die Ein- und Ausgänge ermöglicht. Abbildung 1.13 veranschaulicht dessen Aufbau.

Abbildung 1.13 Bezeichnung von PAL- und GAL-Schaltkreisen (National Semiconductor)

Hinweis:
Die Abbildung veranschaulicht das Bezeichnungssystem der Fa. National Semiconductor bis in die Einzelheiten. Die ersten 4 Bestimmungen (Technologie, Eingangszahl, Ausgangs-Typ, Ausgangszahl) werden von einigen anderen Herstellern genau so angegeben; sie werden auch in der allgemeinen (hersteller-unabhängigen) Literatur viel verwendet. Die drei am weitesten rechts stehenden Angaben (Geschwindigkeit/Verlustleistung, Gehäusetyp, Temperaturbereich) können wir zunächst übergehen.

2. Universelle und programmierbare Logik mit Standardschaltkreisen

2.1. Multiplexer und Decoder

Insbesondere Multiplexer, manchmal aber auch Decoder, sind in der Vergangenheit gern als "Logikersatz" verwendet worden. Obwohl seit dem Erscheinen der GALs an sich veraltet, können solche Lösungen aber auch in neuerer noch vorkommen (ein naheliegender Grund: man muß nicht programmieren, braucht also weder Programmiergerät noch Entwicklungssoftware, und wenn man mit einem Multiplexer auskommt...). Abbildung 2.1 veranschaulicht das Prinzip am Beispiel eines binären Volladdierers.

Abbildung 2.1 Realisierung eines binären Volladdierers

Rekonstruktion der Wahrheitstabelle

1. beim Decoder oder beim nur mit Festwerten beschalteten Multiplexer: Die binär codierten Werte der an den Adreßeingängen anliegenden Signale untereinanderschreiben (z. B. 000, 001, 010 usw. bis 111). Alle diese Wertkombinationen prüfen und die jeweils zugehörige Ausgangsbelegung (0 oder 1) hinschreiben. Fertig.
2. ist der Multiplexer mit einem weiteren Eingangssignal beschaltet, den gemäß Punkt 1. erstellten Teil der Wahrheitstabelle nochmals daruntersetzen. Das Signal an den Dateieingängen wird in einer weiteren, links außen angefügten Spalte dargestellt (in der ersten Hälfte der Zeilen: 0, in der zweiten Hälfte: 1). Dann werden die Dateneingänge nacheinander untersucht. Einen anliegenden Festwert tragen Sie unmittelbar in die Ausgangsspalte ein (in beiden Hälften der Wahrheitstabelle). Ein direkt anliegendes Signal wird direkt übernommen, ein negiert anliegendes negiert

(0 → 1; 1 → 0).

Alternative (Abbildung 2.2):
Schreiben Sie zunächst die Signale hin, die an die Auswahleingänge angeschlossen sind. Rechts daneben kommt das Signal, mit dem die Dateneingänge beschaltet sind. In der Wahrheitstabelle haben Sie dann für jeden Dateneingang jeweils zwei mögliche Belegungen. (Im Beispiel wählen d, c, b den jeweiligen Dateneingang aus.) Dann müssen Sie nur noch nachsehen: Ist der betreffende Dateneingang fest mit 0 oder 1 beschaltet, so erscheint in beiden Fällen der Festwert am Ausgang. Ist er mit Eingang a beschaltet, so müssen Sie die a-Werte direkt als Ausgangswerte übernehmen; ist er mit /a beschaltet, müssen Sie die a-Werte invertieren.

Abbildung 2.2 Multiplexer, als wahlfreie Logik beschaltet

2.2. Adressierbare Speicher

Adressierbare Speicher haben den Vorteil, daß man mit ihnen jede beliebige Schaltfunktion (in den Grenzen der gegebenen Adreßeingänge) verwirklichen kann. 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" AT die Adreßdecodierung auf dem Motherboard teilweise mit Schottky-TTL-PROMs realisiert).

Die Abbildung 2.3 veranschaulicht einige Beispiele.

Abbildung 2.3 PROMs als universelle Logikzuordner

Hinweise:

1. Beim PROM ist es vollkommen gleichgültig, wie kompliziert die kombinatori-sche Zuordnung ist.
2. PROMs werden gelegentlich auch dazu genutzt, an sich beliebige Funktionen der "Restlogik" aufzunehmen. (Es gibt Entwickler, die sogar ein Zweifach-UND im PROM realisieren. Warum auch nicht, wenn die Anschlüsse ohnehin noch frei sind und ansonsten ein weiterer Schaltkreis erforderlich wäre?) - *Aber Vorsicht:*

Speicher als Sammlung verschiedener Schaltfunktionen
Auf eine besondere Überraschung müssen wir uns einstellen, wenn in einem Speicher, z. B. in einem PROM, mehrere unabhängige Schaltfunktionen zu verwirklichen sind (so kann man in einem 2k · 4-PROM bis zu 4 verschiedene Schaltfunktionen unterbringen, wobei jede von beliebigen der insgesamt 11 Adreßeingänge abhängen kann). Nehmen wir an, die Schaltfunktion an Ausgang 0 hängt von den Eingängen A0...A3 ab, jene an Ausgang 1 von den Eingängen A7...A11. Wir messen (mittels Oszilloskop) an Ausgang 0. An den Eingängen A0...A3 ändert sich nichts. Wor messen aber gelegentlich Signale, die wie Störungen aussehen (Nadeln, kurze Impulse usw.). Die Ursache? - Schaltvorgänge an Eingängen, die zu anderen Schaltfunktionen gehören!

Erklärung: wenn sich die Eingangsbelegung ändert, so finden im gesamten Speicher-komplex Umschaltvorgänge statt (dem Speicher ist es schließlich vollkommen gleichgültig, wofür sein Inhalt verwendet wird; die Adreßdecodierung wirkt auf die gesamte Speichermatrix). Diese Umschaltvorgänge äußern sich an den Ausgängen durch Signalwechsel, die oft wie Störungen aussehen.

3. PALs (Programmable Array Logic)

3.1. Kombinatorische PALs

Die Abbildungen 3.1 und 3.2 geben einen Überblick über gängige PALs, die ausschließlich kombinatorische Verknüpfungen enthalten.

Abbildung 3.1 Kombinatorische PALs in 20-Pin-Gehäusen (National Semiconductor)

Abbildung 3.2 Kombinatorische PALs in 24-Pin-Gehäusen (National Semiconductor)

Abbildung 3.3 veranschaulicht den Aufbau einer einfachen PAL-Struktur genauer. Wir erkennen eine programmierbare UND- Anordnung, der ausgangsseitig ODER-Gatter fest nachgeschaltet sind.

Abbildung 3.3 PAL 16H2 (National Semiconductor)

Ausgänge
Manche PALs haben "gewöhnliche" Logikausgänge (bei "H"-Typen direkt, bei "L"- Typen invertiert wirkend). Andere Typen verfügen hingegen über Tri-State-Ausgänge. Die Aufschaltsteuerung ist bei kombinatorischen PALs ebenfalls über die UND-Gatter-Anordnung verwirklicht (der Schaltungsentwerfer kann also Aufschalterlaubnis-Bedingungen (Enables) für jeden derartigen Ausgang als programmierbare UND-Verknüpfung (Produktterm) von Eingangsbelegungen angeben. Manche (gelegentlich auch alle) Ausgänge sind in die UND-Gatter-Matrix zurückgeführt (feedback) und können so in die Bildung von Produkttermen mit einbezogen werden. Abbildung 3.4 zeigt ein Beispiel für eine solche PAL-Struktur.

Abbildung 3.4 PAL 16L8 (National Semiconductor)

Komplementierung (Invertierung)
Es gibt PAL-Typen, die zwischen dem Ausgang des ODER-Gatters und dem entsprechenden Schaltkreis-Pin ein Antivalenzgatter (XOR) enthalten. Dessen zweiter Eingang ist auf einen Festwert (0 oder 1) programmierbar, so daß der Ausgang wahlweise direkt oder invertierend wirkt. Beispiel: PAL 20P8.

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

3.2. Sequentielle (Registered) PALs

Die Abbildungen 3.5 und 3.6 geben einen Überblick über gängige PALs, deren Ausgänge über Flipflops geführt sind. Diese Flipflops werden durch einen gemeinsamen Takt gesteuert und bilden somit praktisch ein Register (deshalb auch die Bezeichnung "Registered PALs").

Abbildung 3.5 Sequentielle (Registered) PALs in 20-Pin-Gehäusen (National Semiconductor)

Abbildung 3.6 Sequentielle (Registered) PALs in 24-Pin-Gehäusen (National Semiconductor)

Abbildung 3.7 zeigt nähere Einzelheiten einer solchen PAL-Struktur.

Abbildung 3.7 PAL 20R8 (National Semiconductor)

Ausgänge
Die Flipflop-Ausgänge sind über Tri-State-Treiberstufen auf die Schaltkreisanschlüsse geführt.

Rückführungen
Die Flipflopausgänge sind auf die UND-Gatter-Matrix zurückgeführt und können so in die Bildung von Produkttermen einbezogen werden (Abbildung 3.8).

Abbildung 3.8 Einzelheit: Flipflop mit ungenutzter und mit genutzter Rückführung (National Semiconductor)

Erzwungenes Laden (Output Register Preload)
Um einen fertig programmierten PAL-Schaltkreis ausprüfen zu können, lassen sich die Flipflops in einem besonderen Modus über die von außen angesteuerten Ausgangs-Anschlüsse laden. Manche Schaltkreise haben einen besonderen Anschluß, um diesen Modus zu erzwingen; bei anderen muß an einem bestimmten Anschluß ein Impuls höherer Spannung (z. B. 18 V) angelegt werden.

Rücksetzen
Beim Anlegen der Versorgungsspannung werden alle Flipflops auf einen definierten Wert (im Einzelnen typabhängig) gesetzt. Achtung: Während die Versorgungsspannung (V_{CC}) ansteigt, sollte der Takt so zeitig wie möglich einen festen Spannungswert annehmen. Während die Versorgungsspannung ansteigt, werden Tri-State-Ausgänge hochohmig gehalten.

Asynchrone Flipflops
Es gibt PAL-Typen, die asynchron setz- bzw. rücksetzbare Flipflops enthalten. Abbildung 3.9 zeigt ein Beispiel (es ist einer der 8 Ausgänge genauer dargestellt).

Abbildung 3.9 PAL 16RA8 (National Semiconductor)

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

Kennwerte (Schottky-TTL)

- Durchlaufverzögerung: 25 ns,
- Taktfrequenz bei Nutzung von Rückführungen: 37 MHz,
- Taktfrequenz ohne Nutzung von Rückführungen: 50 MHz.

Kennwerte (ECL)

- Durchlaufverzögerung: 4...6 ns,
- Taktfrequenz: 117...200 MHz.

4. GALs (Generic Array Logic)

Im Gegensatz zu PALs sind GALs elektrisch programmierbar (und löschbar). Sie enthalten ausgangsseitig vielseitig programmierbare Funktionsblöcke (Output Logic Macrocells OLMCs), so daß man wahlweise sowohl kombinatorische als auch sequentielle Schaltungen verwirklichen kann. Auf Grund dieser Vielseitigkeit kommt man mit einem - im Vergleich zu PALs - geringeren Schaltkreissortiment aus, um die vielfältigen Wünsche der Hardware-Entwickler zu erfüllen. Abbildung 4.1 vermittelt einen Überblick.

Abbildung 4.1 Marktübliche GALs (National Semiconductor)

Die Abbildungen 4.2 bis 4.4 zeigen nähere Einzelheiten. Abgesehen von der Vielseitigkeit und vom anderen Programmierverfahren bieten die GALs dem Schaltungsentwerfer dieselben Möglichkeiten wie entsprechend ausgewählte PALs (Tabelle 4.1).

Abbildung 4.2 GAL 16V8 (National Semiconductor)

Abbildung 4.3 Aufbau einer OLMC-Makrozelle (National Semiconductor)

Abbildung 4.4 Konfigurationsmöglichkeiten einer OLMC-Makrozelle (National Semiconductor)

GAL 16V8 ersetzt folgende PALs	GAL 20V8 ersetzt folgende PALs

10H8	12H6	14H4	16H2	14H8	16H6	18H4	20H2
16H8	16R4	16RP4	10L8	20H8	20R4	20RP4	14L8
12L6	14L4	16L2	16L8	16L6	18L4	20L2	20L8
16R6	16RP6	10P8	12P6	20R6	20RP6	14P8	16P6
14P4	16P2	16P8	16R8	18P4	20P2	20P8	20R8
16RP8				20RP8			

Tabelle 4.1 Ersetzen von PALs durch GALs (nach: Farnell GmbH)

Kennwerte

- Durchlaufverzögerung: 25...30 ns,
- Taktfrequenz bei Nutzung von Rückführungen: 22 MHz,
- Taktfrequenz ohne Nutzung von Rückführungen: 33 MHz.

5. Komplexe programmierbare Logik

Komplexe programmierbare Logikschaltkreise haben - im Vergleich zu "gewöhnlichen" PALs und GALS - komplexere Makrozellen (mehr Funktionen, flexibler programmierbar) und vielseitiger (wahlweise als Eingänge, als Ausgänge oder bidirektional) nutzbare Schaltkreisanschlüsse. Manche Typen bieten zudem kombinatorische Verknüpfungsmöglichkeiten, die über einfache UND-ODER-Strukturen hinausgehen. Wir wollen den Aufbau derartiger Schaltkreise anhand von Beispielen veranschaulichen.

Intel-EPLDs
Die EPLDs von Intel sind in CMOS-Technologie ausgeführt. Sie sind elektrisch programmierbar und mittels UV-Licht löschbar. (Voraussetzung: Glasfenster im Schaltkreisgehäuse. Stichwort: EPROM-Technologie.) Die Abbildungen 5.1 bis 5.3 geben anhand von 3 Beispielen einen Einblick in das Typenspektrum. (Im Anschluß gehen wir kurz auf einige Besonderheiten dieser Schaltkreise ein.)

Hinweis:
Die Schaltkreise sind nicht mehr am Markt. Sie eignen sich aber nach wie vor gut als Beispiele, die zugleich einen Eindruck von der Entwicklungsgeschichte vermitteln. Moderne CPLDs haben mehr Zellen, höher entwickelte Verdrahtungsressourcen, sind schneller usw. Die Prinzipien haben sich aber kaum geändert.

Abbildung 5.1 EPLD 5AC312 (Intel)

Abbildung 5.2 EPLD 5C032 (Intel)

Abbildung 5.3 EPLD 5C180 (Intel)

5AC312

Der Schaltkreis hat 8 fest als Eingänge konfigurierte und 12 freizügig konfigurierbare Signalanschlüsse. Die 8 Eingänge sind über individuell programmierbare Register-Zellen geführt. Abbildung 5.4 zeigt Einzelheiten.

Abbildung 5.4 EPLD 5AC312: Struktur eines Eingangs (oben) und einer E-A-Makrozelle (darunter; Intel)

Programmierbar ist jeweils:

- die Nutzung oder Nichtnutzung (flow-through) des Speicherkreises,
- das Verhalten des Speicherkreises (Latch oder D-Flipflop),
- die Taktierung des Speicherkreises (entweder vom gemeinsamen Takteingang = Synchronbetrieb oder über eigene Terme aus der Gattermatrix = Asynchronbetrieb).

Kennwerte

- Durchlaufverzögerung: 25...30 ns,
- Taktfrequenz bei Nutzung von Rückführungen: 29...33 MHz,
- Taktfrequenz ohne Nutzung von Rückführungen: 50...66 MHz.

5C032

Der Schaltkreis enthält 8 Makrozellen, die jeweils aus einem PLA-Block und einem E-A-Architekturblock bestehen. Jeder PLA-Block hat 8 UND-Gatter zu je 36 Eingängen. Angeschlossen an jedes Gatter sind: (1) die 10 Schaltkreis-Eingänge sowie (2) die 8 Rückführungen aus allen Makrozellen. Alle Signale sind sowohl direkt als auch invertiert anschaltbar (Abbildung 5.5). Der E-A-Architekturblock enthält ein D-Flipflop, dessen Nutzung wiederum programmierbar ist (Abbildung 5.6).

Abbildung 5.5 EPLD 5C032: PLA-Block (Intel)

Abbildung 5.6 EPLD 5C032: E-A-Architekturblock (Intel)

Kennwerte

- Durchlaufverzögerung: 30 ns,
- Taktfrequenz bei Nutzung von Rückführungen: 28 MHz,
- Taktfrequenz ohne Nutzung von Rückführungen: 43 MHz.

5C180

Kennzeichnend für dessen Schaltkreis ist die Anzahl und die Flexibilität der Makrozellen (beispielsweise kann der Speicherkreis in einer solchen Zelle als D-Flipflop, als T-Flipflop, als JK-Flipflop usw. betrieben werden). Zur Ansteuerung der 48 Makrozellen können insgesamt 480 Produktterme gebildet werden. Die Makrozellen sind in 4 Quadranten angeordnet, wobei für jeden Quadranten ein eigener Takt vorgesehen ist. Die Makrozellen eines jeden Quadranten sind über je einen lokalen Bus untereinander verbunden. Weiterhin ist ein globaler Bus vorgesehen, um die Quadranten untereinander zu verbinden und um die Eingangssignale zu den Makrozellen durchzuschalten. Abbildung 5.7 zeigt die Anschaltung einer Makrozelle an die UND-Gatter-Matrix. Die Abbildungen 5.8 bis 5.12 veranschaulichen die verschiedenen Nutzungsweisen.

Abbildung 5.7 EPLD 5C180: Makrozelle (Intel)

Abbildung 5.8 EPLD 5C180: Kombinatorische Nutzung eines E-A-Anschlusses (Intel)

Abbildung 5.9 EPLD 5C180: Betrieb als D-Flipflop (Intel)

Abbildung 5.10 EPLD 5C180: Betrieb als T-Flipflop (Intel)

Abbildung 5.11 EPLD 5C180: Betrieb als JK-Flipflop (Intel)

Abbildung 5.12 EPLD 5C180: Betrieb als RS- (SR-) Flipflop (Intel)

Kennwerte

- Durchlaufverzögerung: 70 ns,
- Taktfrequenz bei Nutzung von Rückführungen: 16 MHz,
- Taktfrequenz ohne Nutzung von Rückführungen: 20 MHz.

5.2. Cypress-PLDs und PALs

Die Fa. Cypress Semiconductor bietet ein Sortiment komplexer programmierbarer CMOS-Schaltkreise an. Die Programmierung erfolgt - vergleichbar zu den Intel-Typen - unter Nutzung der EPROM-Technologie. Abbildung 5.13 veranschaulicht einen solchen Schaltkreis.

Abbildung 5.13 Asynchronous Registered EPLD CY7C331 (Cypress)

Wir erkennen, daß es - auf der Ebene des Blockschaltbilds - keine grundsätzlichen Unterschiede zu den Typen gibt, die wir bisher kennengelernt haben. Abbildung 5.14 zeigt die Struktur einer Makrozelle; Abbildung 5.15 veranschaulicht die verschiedenen Nutzungsweisen. Ein weiteres Kennzeichen der Cypress-Schaltkreise besteht darin, daß aus jeweils zwei Makrozellen über einen programmierbaren Multiplexer ein Eingang ausgewählt und so der UND-Gatter-Matrix zugeführt werden kann (Abbildung 5.16).

Abbildung 5.14 EPLD CY7C331: Struktur einer Makrozelle (Cypress)

Abbildung 5.15 EPLD CY7C331: Nutzungsweisen einer Makrozelle (Cypress)

Abbildung 5.16 EPLD CY7C331: Eingangsauswahl (Shared Input Multiplexer; Cypress)

Kennwerte

- Durchlaufverzögerung: 20 ns,
- Taktfrequenz bei Nutzung von Rückführungen: 16 MHz,
- Taktfrequenz ohne Nutzung von Rückführungen: 20 MHz.

Die Abbildungen 5.17 und 5.18 veranschaulichen einen komplexen PLD-Schaltkreistyp, dessen Auslegung einem FPGA nahekommt. Die Schaltkreise der FLASH370- Serie sind elektrisch programmier- und löschbar. Die Schaltkreise enthalten mehrere Logikblöcke mit UND-ODER-Gattermatrizen sowie Makro- und E-A-Zellen. Diese Logikblöcke werden durch eine programmierbare Verbindungsmatrix (Programmable Interconnect Matrix PIM) untereinander und mit den Eingangs-Anschlüssen verbunden (vgl. auch den Intel-Typ 5C180).

Abbildung 5.17 EPLD CY7C374 (Cypress)

Abbildung 5.18 EPLD CY7C374: Struktur eines Logikblocks (Cypress)

Kennwerte

- Durchlaufverzögerung (vom Eingang zu einem kombinatorischen Ausgang): 12 ns,
- Taktfrequenz ohne Nutzung von Rückführungen: 100 MHz,
- Taktfrequenz bei Nutzung von internen Rückführungen: 66 MHz,
- Taktfrequenz bei Nutzung von externen Rückführungen: 50 MHz.

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 bzw. Makrozellen eingebettet sind. Die einzelnen Ausführungen unterscheiden sich neben technologischen Besonderheiten im wesentlichen in der Komplexität der Funktionsblöcke und im Programmierverfahren.

6.1. Texas-Instruments-FPGAs

Es handelt sich um CMOS-Schaltungen, die nach dem Aufschmelzprinzip (Antifuse) programmiert werden. Die Programmierung ist also nicht wieder löschbar. Abbildung 6.1 veranschaulicht die Anordnung von Funktionsblöcken in der Verbindungsmatrix. Je nach Baureihe bzw. Schaltkreistyp gibt es verschiedene Arten von Funktionsblöcken (Logic Modules). Die Reihe TPC10 hat lediglich kombinatorische Funktionsblöcke (Abbildung 6.2). Hingegen sind die Schaltkreise der Reihe TPC12 abwechselnd mit kombinatorischen und sequentiellen Funktionsblöcken belegt (Abbildung 6.3).

Abbildung 6.1 Funktionsblöcke in der Verbindungsmatrix eines FPGAs (Texas Instruments)

Abbildung 6.2 FPGA-Baureihe TPC10: ein Funktionsblock (Texas Instruments)

Abbildung 6.3 Funktionsblöcke der FPGA-Baureihe TPC12 (Texas Instruments)

Achten Sie darauf, wie der Speicherkreis verwirklicht ist: nämlich als Hintereinanderschaltung zweier Selbsthaltekreise (Latches), die durch rückgekoppelte Multiplexer gebildet werden.

Kennwerte

- Durchlaufverzögerung: je Funktionsblock 5...10 ns.,

- Taktfrequenz: maximal 100 MHz,
- nutzbare Gatter: 1200...6000.

6.2. Cypress-FPGAs

Die FPGAs der pASIC380-Baureihe werden in CMOS-Technologie gefertigt und nach einem Antifuse-Prinzip programmiert. Abbildung 6.4 zeigt die Anordnung von Logikzellen in einem solchen Schaltkreis; Abbildung 6.5 veranschaulicht die Struktur der Logikzellen.

Abbildung 6.4 FPGA-Baureihe pASIC380: Anordnung der Verbindungskanäle und Logikzellen (Cypress)

Abbildung 6.5 FPGA-Baureihe pASIC380: Struktur einer Logikzelle (Cypress)

Kennwerte

- Durchlaufverzögerung: je Logikblock 3...8 ns,
- Taktfrequenz bei schaltkreisinterner Taktierung über 100 MHz, zwischen zwei Schaltkreisen bis 85 MHz,
- nutzbare Gatter: 1000...4000.

6.3. Altera-FPGAs

Die Schaltkreise der FLEX-Serie werden in CMOS-Technologie gefertigt. Die Konfigurationsangaben (Programmierung der Verbindungen und der Funktionen in den einzelnen Zellen) werden in SRAM-Speicherzellen gehalten. Dies gibt höchste Flexibilität (Änderbarkeit während des Betriebs), erfordert aber ein Laden nach dem Einschalten, z. B. aus einem EPROM. Abbildung 6.6 gibt einen Überblick über einen solchen Schaltkreis.

Abbildung 6.6 FPGA-Schaltkreis FLEX EPF81188 (Altera)

Die logischen Funktionsblöcke bestehen aus einzelnen Logikelementen (LEs), von denen jeweils 8 zu einem LAB (Logic Array Block) zusammengefaßt sind. Je ein LAB befindet sich in einem Knotenpunkt der Verbindungsmatrix. Abbildung 6.7 zeigt die Struktur eines Logikelements.

Abbildung 6.7 FPGA-Schaltkreis FLEX EPF81188: Logikelement (LE; Altera)

Beachten Sie, daß man für die Kombinatorik eine etwas ungewöhnliche Lösung gewählt

hat: einen ladbaren Zuordner-Speicher (Lookup Table LUT), der je nach Inhalt eine beliebige Verknüpfung von 4 Variablen verwirklichen kann. Das Flipflop kann wahlweise als D-, T-, JK- oder RS-Flipflop genutzt werden. Zwei zusätzliche Signalwege (CARRY und CASCADE) erlauben schnelle Verbindungen zwischen benachbarten Logikelementen ohne Nutzung der Verbindungsmatrix (wichtig für Zähler, Schieberegister, ALUs usw.).

Abbildung 6.8 zeigt, wie ein Logic Array Block (LAB) aus 8 Logikelementen aufgebaut ist. (Der LAB ist praktisch die kleinste Einheit - ein Funktionsmodul - in der Matrixstruktur.) Der lokale Verbindungsweg (Local Interconnect) ist an die jeweilige Zeile der Verbindungsmatrix bzw. an besondere (decdicated) Eingänge angeschlossen. Er versorgt die Logikelemente im wesentlichen mit Takt- und Rücksetzsignalen.

Abbildung 6.8 FPGA-Schaltkreis FLEX EPF81188: Local Array Block (LAB; Altera)

Kennwerte

- Durchlaufverzögerung: je Logikelement 3...6 ns,
- Taktfrequenz: über 70 MHz,
- nutzbare Gatter: 4000...24000.

6.4. Xilinx-LCAs

Die LCA-Schaltkreise (LCA = Logic Cell Array) von Xilinx werden in CMOS-Technologie gefertigt. Ähnlich wie bei den Altera-Typen werden die Programmierangaben in RAM-Zellen gehalten. LCA-Schaltkreise müssen also bei jeder Inbetriebnahme geladen werden (und sind während des Betriebs umladbar). Der einzelne Funktionsblock heißt hier CLB (Configurable Logic Block). Abbildung 6.9 zeigt die Anordnung eines CLBs in der Verbindungsmatrix; Abbildung 6.10 veranschaulicht dessen Struktur.

Abbildung 6.9 Ein CLB in der Verbindungsmatrix (Xilinx)

Abbildung 6.10 Struktur eines CLB (Xilinx)

Die drei mit "LOGIC FUNCTION..." bezeichneten Blöcke sind ladbare RAM-Zuordner, ähnlich wie bei Altera. Sie werden hier als Funktionsgeneratoren bezeichnet. Eine

Besonderheit: sie können auch als "echte" RAMs genutzt werden, also um tatsächlich Daten zu speichern. Hierfür gibt es drei verschiedene wählbare Konfigurationen:

- 2 RAMs der Organisation "16 · 1",
- 1 RAM der Organisation "32 · 1",
- 1 RAM der Organisation "16 · 1" sowie ein Logikzuordner mit 5 Eingängen.

Kennwerte

- Durchlaufverzögerung in einem CLB: 4...8 ns,
- Taktfrequenz über 40 MHz,
- nutzbare Gatter: 2000...20000.

7. State Machines

State Machines kann man mit an sich beliebigen PLDs aufbauen. Wesentlich ist, ob die nutzbare Kombinatorik ausreicht, die erforderlichen Zustandswechsel zu verwirklichen. Es gibt PLDs, deren Makrozellen ausdrücklich für State-Machine-Anwendungen vorgesehen sind (das betrifft im besonderen die "synchrone" Taktierung).

Die Kompliziertheit der Zustandswechsel spielt keine Rolle, wenn man eine ROM-Struktur als Zuordner verwendet. Um mit möglichst wenig Schaltkreisen auszukommen, werden Registered PROMs und State Machine PROMs angeboten. Letzere enthalten auch die - programmierbaren - Rückführungen der Zustandscodierung auf die Adreßeingänge. Die Abbildungen 7.1 bis 7.3 veranschaulichen entsprechende Beispiele.

Abbildung 7.1 Registered PROM CYC245A (Cypress)

Abbildung 7.2 Registered PROM CYC277 (Cypress)

Abbildung 7.3 State Machine PROM CYC259 (Cypress)

Kennwerte

- CYC245A: Verzögerung Adresse zu Daten: 15...45 ns,
- CYC259: Zykluszeit 10, 12, 15 ns,
- CYC277: Adreß-Vorhaltezeit: 30 ns; Verzögerung Takt zu Daten: 15...25 ns.