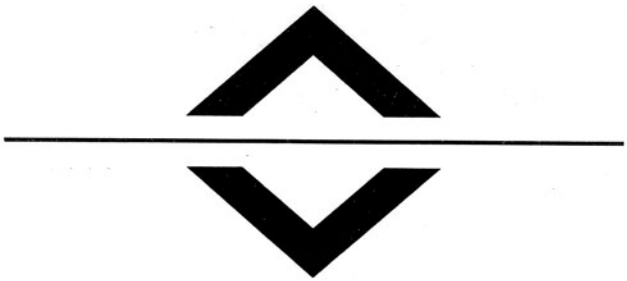


NUCLEAR INTERFACE

Bedienungsanleitung

PAL-Programmer PPR-2

Bedienungsanleitung

PAL - Programmer Karte für Apple II

Dipl. Phys. M. ZAHN
Baldeplatz 1
8 München 5

I N H A L T

1.) Allgemeine Hinweise

Bla Bla:

Vorteile PALs
- sparen ICs (chip count)
- Platz auf der Leiterplatte
- leichtere Entflechtung (Ein, Ausgänge frei prog)
- vereinfachte Lagerhaltung
- umgehen Beschaffungsschwierigkeiten
- Lese, Nachbauschutz

Apple Pal Prog:

- preiswerter Einstieg
- Programmer und Hostrechner in einem
- besonders kurzer Entwicklungszyklus

Wir sind dankbar für Kritik und Verbesserungsvorschläge zu Software und Hardware.

Garantie: 6 Monate Teile und Arbeit, nicht eingeschlossen sind Schäden durch unsachgemässe Behandlung, wie z.B Stecken der Karte unter Stromversorgung.

2.) Installation

Achtung:

Installieren sie die Karte nur, wenn die Stromversorgung Ihres Computers abgeschaltet ist!

Die Karte kann grundsätzlich in jeden der Slots 1 bis 7 gesteckt werden. Es sollte dabei darauf geachtet werden, daß die Wechselfassung für das PAL IC gut zugänglich ist, also rechts neben der Karte freier Platz zur Verfügung steht. Dies ist gewöhnlich bei Slot 7 und, wegen der kurzen Disk-Controller Karte, auch bei Slot 5 der Fall.

Wenn die Karte auf eine Extenderkarte gesteckt wird, ist zwar die Wechselfassung sehr gut zugänglich, es ist aber nötig, die Karte dann ausgiebig zu testen, da bekanntlich beim Betrieb mit Extender wegen der zusätzlichen Leitungslängen Probleme auftreten können.

Falls die PALs recht häufig gewechselt werden sollen ist es empfehlenswerter die 24 pol. Fassung zu verlängern. Man benutzt am besten ein Flachbandkabel mit einem DIL-Stecker an der einen, und einer neuen Wechselfassung an der anderen Seite. Verlängerungen von bis zu 30 cm sind ohne Probleme möglich.

Die Programmierspannung für die PALs ist direkt von der 12 V Versorgung des APPLE Netzteils abgeleitet. Es ist deshalb auf alle Fälle zu empfehlen, die Spannung des Netzteils mit einem Digitalvoltmeter zu überprüfen. Bei einer Abweichung von mehr als 0.3 V sollte das Netzteil überprüft, oder eine externe 12V Versorgung verwendet werden. Während erfahrungsgemäß die Karte auch bei erheblicher Unterspannung noch funktionieren dürfte, besteht bei zu hoher Spannung Gefahr für Ihre PAL ICs.

Ein theorethischer Nachteil der Karte soll nicht verschwiegen werden. Das Verifizieren der gebrannten Sicherungen, geschieht nur bei einer Versorgungsspannung von ca. 5V, und nicht wie im Datenblatt verlangt bei 6V und bei 4.5V. Dies ist ein Zugeständnis an die Forderung nach einem möglichst günstigen Preis für die Karte. Da aber in ausgiebigen Tests mit PALs verschiedener Hersteller keine daraus resultierenden Fehler aufgetreten sind, sind wir sicher, daß dies keine Einschränkung bedeutet.

Alle anderen Spezifikationen der Hersteller bezüglich der Programmierspannungen, und des Timing, werden von der Karte eingehalten.

Die LED rechts oben an Ihrer Karte leuchtet immer dann auf, wenn die Karte selektiert wird. Da das Programmieren der PALs nur wenig Zeit in Anspruch nimmt, sieht man die LED meist nur kurz aufleuchten.

Ein Hardware RESET schaltet die Karte ab.

Achten sie immer darauf, daß sich der Schalter zwischen den 20 pol. und den 24 pol. PALs in der richtigen Stellung befindet, bei falscher Stellung, kann das IC, nicht aber die Karte zerstört werden.

Ebenfalls führt ein Programmierversuch an einem in der falschen Richtung eingesetzten IC zu dessen Zerstörung. Die folgende Zeichnung erklärt, wie das IC richtig eingesetzt wird:

von oben:

20 pol. PAL 24 pol. PAL

Wenn die Karte in den Computer eingesteckt ist, sieht das ganze so aus:

20 pol. PAL

24 pol. PAL

3.) Software

3.1 Allgemeines:

Bei der mitgelieferten Software handelt es sich um den auf APPLESOFT BASIC umgeschrieben BASIC PALASM 20 von MMI. Das Programm wurde um die für das Programmieren der PAL6 notwendigen Maschinenroutinen erweitert, und auch sonst in Hinblick auf Benutzerfreundlichkeit überarbeitet.

Die Rückseite der Diskette kann unter dem CPM Betriebssystem gelesen werden, und enthält den von MMI frei erhältlichen FORTRAN Sourcecode für den PALASM 20 und PALASM 24.

Die FORTRAN Version von PALASM ist auf jeden Fall komfortabler und vielseitiger als die BASIC Version. Es ist aber aus Urheberrechtlichen Gründen nicht möglich diese Programme in compilierter Form mitzuliefern. Falls sie mit der angebotenen Software nicht zufrieden sind, müßten sie sich den MICRO SOFT Fortran Compiler erwerben, mit dem die PALASM Sourcefiles problemlos übersetzbar sind.

In Kapitel 3.4 wird genau beschrieben, wie sie, mit dem FORTRAN PALASM unter CPM erstellte PAL-Codefiles mit den mitgelieferten DOS Programmen in PALs programmieren können. Auf ähnliche Weise lassen sich auch auf anderen Rechnern erstellte Codefiles programmieren.

Da zum gegenwärtigen Zeitpunkt kein BASIC PALASM für 24 pol. PALs verfügbar ist, wird das Hilfsprogramm PALHELP24 mitgeliefert, welches die manuelle Eingabe der zu brennenden Sicherungen im PAL erlaubt, aber nicht in der Lage ist, symbolische Gleichungen zu assemblieren.

Die Praxis zeigt, daß das Eingeben des 'Fuse pattern' von Hand kein wesentlicher Nachteil gegenüber dem Arbeiten mit symbolischen Gleichungen ist. Falls der Benutzer die Möglichkeit hat, den FORTRAN Sourcecode des originalen PALASM 24 zu übersetzen ist dies natürlich dem Arbeiten mit PALHELP24 vorzuziehen. Das Programm PALHELP24 wird dann nur noch verwendet um 24 pol. PALs zu brennen, bzw. zu lesen.

Hinweise:

Falls möglich sollte mit einer 80 Z Karte gearbeitet werden, da der vom Programm ausgegebene Text, insbesondere der 'Fuse Plot' sonst sehr unübersichtlich sind.

Es wird angenommen, das die Druckerkarte in Slot 1 installiert ist.

3.2 APPLESOFT BASIC PALASM20

Es liegt nicht in der Absicht dieser Anleitung, das Arbeiten mit der PALASM Software erschöpfend zu behandeln. Neulingen auf diesem Gebiet empfehlen wir dringend die entsprechende Literatur, und die PAL Datenbücher zu konsultieren bevor sie das erste PAL programmieren.

Als kleine Hilfestellung wird im folgenden ein typischer Entwicklungszyklus beschrieben:

1. Erstellen eines PALASM Source Files mit einem Texteditor (z.B.: EDASM):

   Erste Zeile: PAL - Typ
   Zweite bis vierte Zeile: Kommentar
   Fünfte Zeile: PIN - Liste, PINs durch Blanks getrennt, kann über mehrere Zeilen fortgesetzt werden
   Sechste bzw. weitere Zeile: PAL Gleichungen in der Form Output = Input term
   Letzte Zeile: 'END' String

   Beispiel:

Weitere Beispiele befinden sich auf der mitgelieferten Diskette.

2. Starten sie PALASM20 und geben sie bei Optionselect ein 'A' (für Assemble) ein.

   Nach der Eingabe des Filenamens wird Ihr Textfile assembliert. Wiederholen sie Schritt 1 und 2 bis das Assemblieren ohne Fehler erfolgt.

3. Sie erhalten den Fuseplot durch Eingabe von 'X' (bzw. 'P' für Ausgabe auf den Printer) bei Option Select.

   Kontrollieren sie das Ergebnis durch Vergleich mit dem Innenschaltbild des PALs im Datenblatt.

   Hinweis:

   Phamtom Fuses werden nur bei assemblierten Fusepatterns gezeigt, nicht aber bei von der Disk geladenen Codefiles. Wenn sie also einen erstellten Code abspeichern, und darauf wieder laden, ist die Information, welches die Phantom Sicherungen sind, verlorengegangen. Auf die Programmierung des PALs hat dies keinen Einfluß.

4. Speichern sie das Ergebnis mit 'W' (für Write)

   Wählen sie einen Filenamen, mit dem sie den erstellten Code jederzeit wieder identifizieren können. PALASM20 speichert nur den reinen binären Code für die Programmiersubroutine, und keinerlei sonstige Information, auch nicht z.B. den PAL Typ.

5. Geben sie bei Option Select 'C' (für Configure) ein.

Das Maschinenunterprogramm von PALASM20 muß den Slot, in dem die PAL - Programmer Karte steckt, mitgeteilt bekommen. Außerdem haben sie die Möglichkeit zu wählen, ob die Maschinenroutine die das PAL programmiert, nach einem Verify Fehler (Meldung: Cannot burn Fuse at ->) abbrechen, oder mit der Programmierung fortfahren soll.

Beantworten sie die beiden Fragen und warten sie bis das konfigurierte Programm auf der Diskette abgespeichert ist.

Diese Änderungen sind permanent, sie müssen die Option 'C' erst wieder wählen , wenn sie die Konfiguration ändern wollen, also z.B. wenn sie die PAL Programmer Karte in einem anderen Slot installieren.

6. Programmieren sie das PAL mit 'B' (für Burn)

Achten sie darauf, daß der Schalter 20/24 auf 20 auf 20 steht und daß das PAL richtig in der Wechselfassung steckt. Wenn sie sich nicht sicher sind, vergleichen sie nochmals mit den Zeichnungen in Kapitel 2.3

Bei erfolgreicher Programmierung hören sie einen kurzen Ton, und das Programm kehrt wieder zu Option Select zurück.

5.) Leseschutz für PALs

Es ist möglich, Ihre programmierten PALs gegen ein Auslesen Ihrer internen Verdrahtung zu schützen, indem die sogenannte Verify-Protect-Fuse geschmolzen wird. Dies ist nicht mit der PAL Programmer Karte möglich, da hierfür eine höhere Betriebsspannung notwendig ist.

Damit sie aber Ihren PAL - Entwurf gegen Nachahmung schützen können, ist hier eine kleine Schaltung angegeben, die den für das Durchbrennen der Sicherung notwendigen Puls liefern kann. Die Schaltung benötigt eine externe Stromversorgung von 21V / 400 mA.

Kontrollieren sie nach dem Aufbau der Schaltung bei einigen PALs ob sie nach dem Brennen der Verify-Protect-Fuse noch lesbar sind.

Inhaltsverzeichnis

Bedienungsanleitung PAL-Programmer PPR-2

Inbetriebnahme des Systems:

Zum Betrieb des PAL-Programmers ist ein Rechner vom Typ Apple II, Apple IIe oder ein kompatibler Rechner mit zwei Diskettenlaufwerken und 80-Zeichen-Karte sowie 64 K-byte Ram erforderlich. Ein Drucker ist wünschenswert. Die Interfacekarten sollten folgendermaßen angeordnet sein:

Floppyinterface: Slot 6
(Druckerinterface: Slot 1)
80-Zeichen-Karte: Slot 3
PAL-Programmer-Karte: beliebiger freier Slot 1-7

Achtung: Karten dürfen nur bei ausgeschaltetem Rechner eingesteckt oder herausgezogen werden.

Starten des Programmes:

Die Programmdiskette ist in das Laufwerk 1, die Datendiskette in das Laufwerk 2 einzulegen. Nach dem Einschalten des Rechners wird das Programm automatisch geladen und gestartet.

Allgemeine Hinweise zur Bedienung:

Die Bedienung des Systems erfolgt interaktiv: Der Rechner zeigt dem Benutzer zu jeder Zeit, welche Kommandos (Eintastenbefehle) möglich sind und reagiert nicht auf ungültige Eingaben. Die zu betätigenden Tasten sind in der Liste der Kommandos durch ein nachfolgendes "<" kenntlich gemacht. So muß, um z.B. das Kommando "G<et data from PAL" auszuwählen, nur die Taste "G" gedrückt werden (Kleinbuchstaben sind auch zulässig).

Die Programmdiskette ist kopiergeschützt. Es sollte deshalb, um eine Beschädigung so weit wie möglich auszuschließen, der Schreibschutz nicht entfernt werden.

Es werden z.T. Programmteile von der Programmdiskette nachgeladen. Deshalb darf niemals die Programmdiskette aus dem Diskettenlaufwerk 1 entnommen werden.

Der Gebrauch von Accelerator-Karten in Zusammenhang mit den Get-und Burn-Routinen ist kritisch (bei falschem Timing können die PALs beschädigt werden). Eingerichtet und getestet ist nur die Funktion mit der Karte SpeeDEMON.

Beschreibung des Programmes:

Nach dem Einschalten meldet sich das System mit:

```
*************************************************************
*                                                           *
* PAL PROGRAMMER:              (C) nuclear interface GmbH   *
*                                                           *
*     P<alasm                                               *
*     F<use pattern                                         *
*                                                           *
*     E<ditor                                               *
*                                                           *
*     Delet<e files                                         *
*     L<ist files                                           *
*     !<format disk                                         *
*                                                           *
*     D<ate:  1-Feb-85                                      *
*                                                           *
*************************************************************
```

Beschreibung der Kommandos:

L<ist files

Es werden alle Files (Datensätze), die auf der Datendiskette gespeichert sind, auf dem Bildschirm aufgelistet. Das mit aufgelistete Datum gibt Aufschluß darüber, wann der Datensatz erzeugt wurde. (Die Namen von Textfiles enthalten das Suffix ".text", die von Datenfiles ".data".) Zusätzlich wird angezeigt, wieviel Platz auf der Datendiskette noch frei ist.

Delet<e files

Hiermit können Files, die nicht mehr benötigt werden, gelöscht werden.

!<format disk

Eine neue Diskette muß, bevor sie benutzt werden kann, formatiert werden. Sie ist dazu in das Laufwerk 2 einzulegen. Wird eine schon bespielte Diskette benutzt, so geht sämtliche gespeicherte Information verloren!!

D<ate

Es kann das aktuelle Datum eingegeben werden. Das Datum wird auf der Datendiskette gespeichert. Alle erzeugten Datensätze werden ebenfalls mit diesem Datum versehen.

Durch die übrigen Kommandos (P,F,E) gelangt man zu anderen Programmteilen. Die erste Zeile im Bildschirm gibt die Programmebene an, in der man sich gerade befindet (z.B. PALASM:).

Dieser Programmteil enthält ein PAL-Assemblier- und Simulationsprogramm (vgl. (1)). Nach fehlerfreier Assemblierung und Simulation können die zum Brennen eines PALs erforderlichen Daten (Fuse-Matrix, PAL-Typ etc.) auf Diskette gespeichert werden. Eine Ausgabe der Fuse-Matrix auf dem Bidschirm bzw. einem Drucker ist ebenfalls möglich.

Das System meldet sich mit:

```
**********************************************************************
*                                                                    *
* PALASM:                                                            *
*                                                                    *
*     E<rrors, test vectors, view, and list to CONSOLE (printer)     *
*                                                                    *
*     N<ame of text and data file=    unnamed                        *
*     L<ist text                                                     *
*     A<sm text                                                      *
*     asm and S<imulate text                                         *
*     V<iew fuse pattern                                             *
*     D<ata file to disk                                             *
*     Q<uit PALASM:                                                  *
*                                                                    *
**********************************************************************
```

Beschreibung der Kommandos:

E<rrors, test vectors, view, and list to CONSOLE (printer)

Die Fehlermeldungen des PAL-Assemblierprogramms, die Testvektoren des Simulationsprogramms, die Fuse-Matrix und der Quelltext können wahlweise auf dem Bilschirm (Console) oder auf einem Drucker (Printer) ausgegeben werden. Das Kommando E<rrors bewirkt ein Umschalten von CONSOLE auf PRINTER und umgekehrt. Es wird empfohlen, die Testvektoren immer auf dem Drucker auszugeben, da deren Berechnung recht lange dauert und bei Fehlern meist ein anschließender Vergleich mit dem Quelltext nötig ist.

N<ame of text and data file= unnamed

Hiermit wird zunächst der Quelltext bestimmt, der zu assemblieren ist. Im Allgemeinen wird der erzeugte Datensatz den gleichen Namen wie der Quelltext haben. Wird für den Datensatz ein anderer Name gewünscht, so ist dieser Name nach der Assemblierung, jedoch vor der Datenspeicherung einzugeben.

Bildschirm-Editor für Fuse-Pattern:

Der Bildschirm-Editor meldet sich mit:

```
*****************************************************************
* Q<uit,<X>,<->,S<et rect mode, <I>=up,<M>=down,                *
*                          <J>=left,<K>=right, <1>..<4>         *
*                                                               *
*   Slot: 5, PAL part type: PAL16R4  (20-pin)                   *
*                                                               *
*  0 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx                    *
*  1 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx                    *
*  . xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx                    *
*  7 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx                    *
*                                                               *
*  8 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx                    *
*  . xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx                    *
* 15 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx                    *
*                                                               *
*****************************************************************
```

Mit Hife dieses Bildschirm-Editors lassen sich auf einfache Weise Fuse-Pattern erzeugen oder verändern.

Die Fuse-Matrix ist in 4 (bzw. 5 bei 24-pol. PALs) Seiten aufgeteilt. Man gelangt in die entsprechende Seite entweder indem der Cursor entsprechend oft um eine Zeile nach unten (bzw. oben) bewegt wird oder schneller durch Drücken der Tasten <1> - <4> (bzw. <5> bei 24-pol. PALs). Der Cursor kann sowohl über die Tasten <I>,<K>,<J>,<L> als auch (falls vorhanden) über die entsprechenden Cursortasten in alle Richtungen bewegt werden. Ein "x" symbolisiert eine ungebrannte, ein "-" eine gebrannte Fuse. Ein Eintrag in der Fuse-Matrix wird verändert, indem der Cursor an die entsprechende Position bewegt und dann die Tase <X> oder <-> gedrückt wird. Größere, zusammenhängende Bereiche werden geändert, indem zunächst der Cursor in eine Ecke des zu ändernden Bereiches gebracht und das Feld durch <X> oder <-> entsprechend gesetzt wird. Mit S<et rect mode (rectangle mode) wird diese Änderung für die folgenden Cursorbewegungen gespeichert. Wird z.B. der Cursor zunächst nach rechts und anschließend nach oben bewegt, so wird das gesamte Rechteck zwischen Anfangs-und Endposition des Cursors verändert. Durch Betätigen anderer als den Cursortasen oder durch Überschreiten einer Seitengrenze wird dieser Modus wieder aufgehoben. Mit Q<uit gelangt man zurück zum Programmteil FUSE PATTERN:.

PAL-Editor:

Mit dem PAL-Editor können die für die Assemblierung erforderlichen Text-Files (Datensätze) erzeugt werden. Die Vorschriften für einen solchen Text (gültige Symbole etc.) entnehmen Sie bitte dem PAL-Handbuch von MMI (1).

Ein neuer Textfile wird angelegt, indem auf die Frage des PAL-Editors "Input file ? (<ret> for new file)" nur die Return-Taste gedrückt wird. Auf die anschließende Frage "Output file ?" wird der Name eingegeben, unter dem der Text auf der Diskette gespeichert werden soll. Dieser Name darf maximal zehn Zeichen lang sein. Nach richtiger Eingabe erscheint die Kommandozeile: PAL EDITOR: I<nsert, D<elete, M<em, P<age, O<p.page, V<erify, Q<uit.

Beschreibung der Kommandos:

I<nsert

Dieser Modus erlaubt das Einfügen von Zeilen oder einzelnen Zeichen in den Text. Er ist auch zu benutzen, wenn ein Text neu begonnen werden soll. Die Eingabe muß mit <ctrl C> (gleichzeitiges Drücken der Tasten <ctrl> und <C>) abgeschlossen werden.

D<elete

Es können einzelne Zeichen oder auch ganze Zeilen gelöscht werden, indem der Cursor an den Beginn der zu löschenden Stelle gebracht, D<elete aufgerufen, der Cursor auf das erste nicht mehr zu löschende Zeichen (mittels Cursortasten und Return) gebracht und <Ctrl C> gedrückt wird.

M<em

Es wird angezeigt, wieviel Speicherplatz noch zur Verfügung steht und in welcher Zeile des Textes sich der Cursor gerade befindet.

P<age

Der Cursor springt eine Seite vorwärts.

O<p.page

Der Cursor springt eine Seite zurück.

V<erify

Der Bildschirminhalt wird berichtigt und die Kommandozeile angezeigt.

Q<uit

Der PAL-Editor wird verlassen. Das System antwortet mit:

E<xit without update
U<pdate to file "Filename"
R<eturn to edit

R<eturn.. wird benutzt, falls versehentlich Q<uit benutzt wurde. Mit U<pdate wird der geänderte Text auf die Diskette zurückgeschrieben. Hat er denselben Namen wie der Input-Text, so wird letzterer gelöscht. Falls versehentlich falsche Änderungen vorgenommen wurden und der Ausgangstext erhalten bleiben soll, bietet E<xit diese Möglichkeit.

Es wird im Programmteil PAL PROGRAMMER: fortgefahren.

Anmerkung für UCSD-Pascal Besitzer:

Es kann auch der im UCSD-System enthaltene Editor benutz werden. Dieser ist dazu unter dem Namen SYSTEM.EDITOR auf die Programmdiskette zu kopieren. Nach dem Verlassen des Editors wird dann jedoch nicht mehr automatisch im Programmteil PAL PROGRAMMER: fortgefahren, sondern es muß der Befehl I(nitialize bzw. X(ecute PALPROG benutz werden.

Gültige PAL-Typenbezeichnungen:

20-polig:

10H8, 12H6, 14H4, 16H2, 16C1,
10L8, 12L6, 14L4, 16L2, 16L8,
16R8, 16R6, 16R4, 16X4, 16A4

24-polig:

12L10, 14L8, 16L6, 18L4, 20L2,
20C1, 20L10, 20X10, 20X8, 20X4,
20L8, 20R8, 20R6, 20R4

1) PAL Handbook
Monolithic Memories GmbH
Mauerkircherstr. 4
8000 München 80
Tel. 089-984961
Telex 524385

Anhang zur Bedienungsanleitung PAL-Programmer PPR-2

1. Fehler im PALASM

PAL-Assembler-Texte für die PALs

20X4
20X8
20X10
16X8

werden fehlerhaft übersetzt. Der Fehler besteht darin, daß das XOR (:+:) wie ein OR (+) behandelt wird. Dieser Fehler kann umgangen werden, indem eine entsprechende Anzahl von ORs vor dem XOR eingefügt wird, z. B. beim 16X8

```
Y := A + GND + GND + GND :+: B
```

statt

```
Y := A :+: B,
```

wenn GND die Bezeichnung für Pin 20 (Masse) ist.

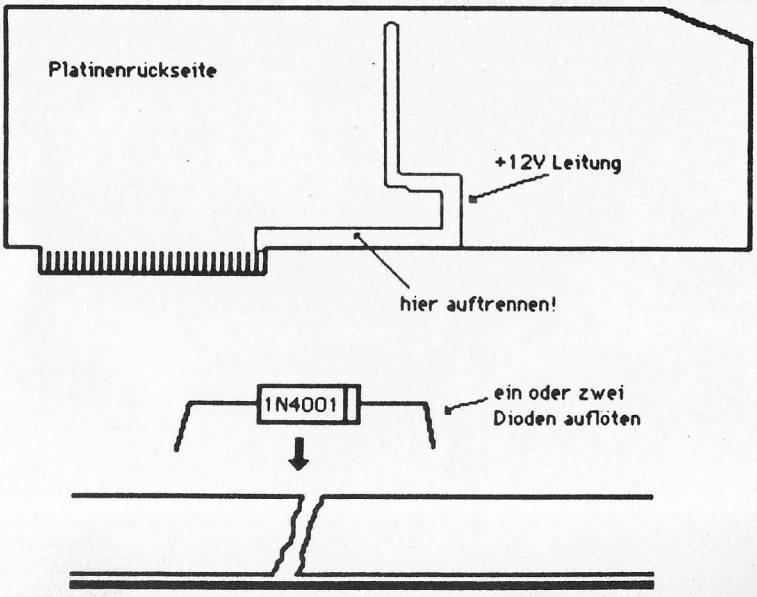
2. Änderung für PALs der Firma TEXAS

Die benötigte Programmierspannung für TEXAS PALs weicht erheblich von der für MMI PALs ab, so daß ein Brennen von TEXAS PALs nur durch Änderung der Hardware möglich ist. Ein Brennen der Security Fuse ist nicht möglich, da hierfür abermals eine andere Programmierspannung erforderlich ist.

Falls Sie TEXAS PALs brennen wollen, gehen Sie bitte folgendermaßen vor:

Stecken Sie ein PAL in die Fassung (als Belastung für die Spannungsquelle) und messen Sie bei eingeschaltetem Rechner die Spannung auf der +12 Volt-Leitung (siehe Skizze) gegen Masse. Masse finden Sie zum Beispiel am Videoausgang des Apple. (Die Karte darf nur bei ausgeschaltetem Rechner gesteckt oder gezogen werden. Achten Sie darauf, daß beim Messen keine Kurzschlüsse verursacht werden.)

Für TEXAS PALs muß die Versorgungsspannung des PAL-Programmers auf 10,5 Volt bis 11,0 Volt reduziert werden. Unterbrechen Sie dazu die +12V Leitung des Programmers auf der Platine gemäß der Skizze. Fügen Sie eine oder zwei in Serie geschaltete Dioden 1N4001 oder ähnlich in die Leitung ein. Die Kathode der Diode muß vom Stecker weg zeigen. Jede Diode reduziert die Versorgungsspannung um ca. 0,8 Volt.



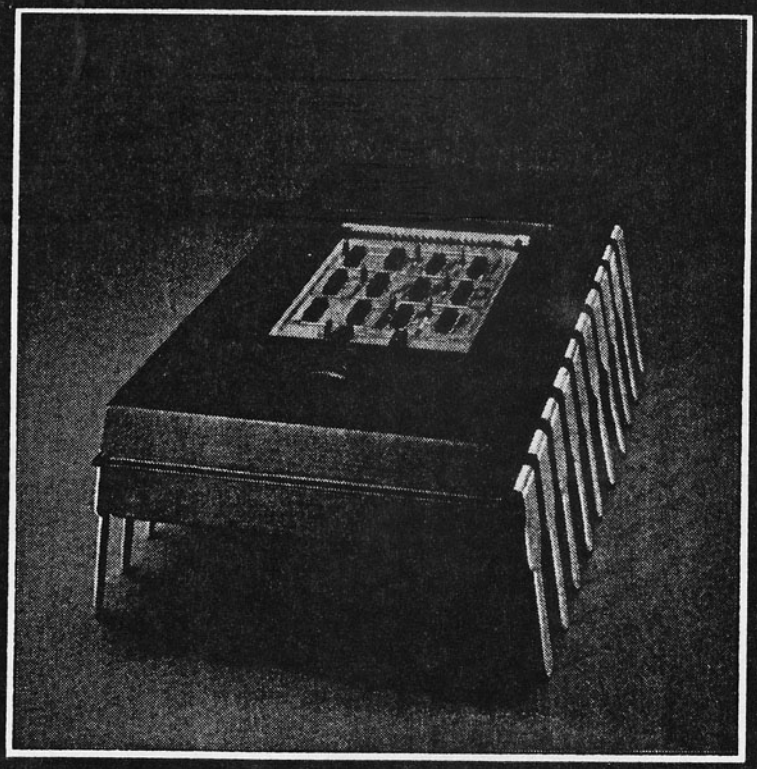
IdeaLogic™

PAL® PROGRAMMABLE ARRAY LOGIC

HAL® HARD ARRAY LOGIC

PLE™ PROGRAMMABLE LOGIC ELEMENTS

DEDICATED LOGIC



Monolithic Memories MMI

Inhalt

IdeaLogic™

ist eine Entwicklungsmethodik, mit der digitale Systeme sofort auf dem Siliziumchip realisiert werden können.
Dies erfolgt mit Hard- und Software nach folgendem Ablauf:

- Defi[illegible]
- Simu[illegible]
- Produkt
- Test

Der Vorteil [illegible] zeit mit der damit verbundenen [illegible] niedrigsten Initialkosten, kürzesten Testzeiten und kompletter Dokumentation im Hause des Anwenders.

In diesem Aufsatz versuchen wir, [illegible] IdeaLogic™ – Entwicklung auf Silizium in verständlicher Form nahe zu bringen.

PAL® PROGRAMMABLE ARRAY LOGIC

Bezeichnung	Eing.	Ausg.	Programmierbare I/O's	Feedback Register	Ausgangs-Polarität	Funktion	Eigenschaften					
							STD	A	B*	–2	–4	CMOS* ECL*
PAL10H8	10	8			AND-OR	AND-OR Gate Array	0/0			0/0		
PAL12H6	12	6			AND-OR	AND-OR Gate Array	0/0			0/0		
PAL14H4	14	4			AND-OR	AND-OR Gate Array	0/0			0/0		
PAL16H2	16	2			AND-OR	AND-OR Gate Array	0/0			0/0		
PAL16C1	16	2			AND-OR/NOR	AND-OR Gate Array	1/1					
PAL20C1	20	2			AND-OR/NOR	AND-OR Gate Array	1/1					
PAL10L8	10	8			AND-NOR	AND-OR Invert Gate Array	0/0			0/0		
PAL12L6	12	6			AND-NOR	AND-OR Invert Gate Array	0/0			0/0		
PAL14L4	14	4			AND-NOR	AND-OR Invert Gate Array	0/0			0/0		
PAL16L2	16	2			AND-NOR	AND-OR Invert Gate Array	0/0			0/0		
PAL12L10	12	10			AND-NOR	AND-OR Invert Gate Array	1/1					
PAL14L8	14	8			AND-NOR	AND-OR Invert Gate Array	1/1					
PAL16L6	16	6			AND-NOR	AND-OR Invert Gate Array	1/1					
PAL18L4	18	4			AND-NOR	AND-OR Invert Gate Array	1/1					
PAL20L2	20	2			AND-NOR	AND-OR Invert Gate Array	1/1					
PAL16L8	16	8	6		AND-NOR	AND-OR Invert Gate Array	3/0	X	X	1/1	X	X
PAL20L8	20	8	6		AND-NOR	AND-OR Invert Gate Array		X	X			
PAL20L10	20	10	8		AND-NOR	AND-OR Invert Gate Array	2/2					X
PAL16R8	16	8		8	AND-NOR	AND-OR Invert Array w/Reg's	3/0	X	X	1/1	X	X
PAL16R6	16	8	2	6	AND-NOR	AND-OR Invert Array w/Reg's	3/0	X	X	1/1	X	X
PAL16R4	16	8	4	4	AND-NOR	AND-OR Invert Array w/Reg's	3/0	X	X	1/1	X	X
PAL20R8	20	8		8	AND-NOR	AND-OR Invert Array w/Reg's		X	X			
PAL20R6	20	8	2	6	AND-NOR	AND-OR Invert Array w/Reg's		X	X			
PAL20R4	20	8	4	4	AND-NOR	AND-OR Invert Array w/Reg's		X	X			
PAL20X10	20	10		10	AND-NOR	AND-OR-XOR Invert w/Reg's	3/2					
PAL20X8	20	10	2	8	AND-NOR	AND-OR-XOR Invert w/Reg's	3/2					
PAL20X4	20	10	6	4	AND-NOR	AND-OR-XOR Invert w/Reg's	3/2					
PAL16X4	16	8	4	4	AND-NOR	AND-OR-XOR Invert w/Reg's	4/1					
PAL16A4	16	8	4	4	AND-NOR	AND-CARRY-OR-XOR Invert w/Reg's	4/1					
PAL10P8	10	8			AND-OR/NOR	AND-OR/NOR Gate Array	0/0	X		0/1		
PAL12P10	12	10			AND-OR/NOR	AND-OR/NOR Gate Array	1/1	X		0/1		
PAL16P8	16	8			AND-OR/NOR	AND-OR/NOR Gate Array	3/0	X		1/1	X	
PAL16FF8	16	8	8	8	AND-OR/NOR	AND-OR/NOR Gate Array w/asyn. FF	3/0					
PAL20FF10	20	10	10	10	AND-OR/NOR	AND-OR/NOR Gate Array w/asyn. FF	3/0					
PAL16RP8	16	8		8	AND-OR/NOR	AND-OR/NOR Gate Array w/Reg's	4/0	X				X
PAL32R16	32	16	16	16	AND-NOR	AND-OR Invert Gate Array w/Reg's	4/0	X				X
PAL64R32	64	32	32	32	AND-NOR	AND-OR Invert Gate Array w/Reg's	3/0					X

ERKLÄRUNG:

STD = Standard
I_{CC} (max): 0 = 90 mA, 1 = 100 mA, 2 = 165 mA, 3 = 180 mA, 4 = 225 mA
t_{PD} (max): 0 = 35 ns, 1 = 40 ns, 2 = 50 ns

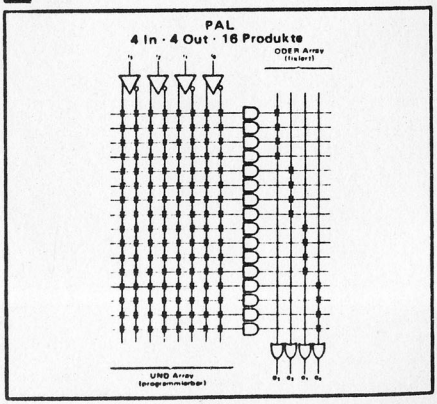
A = High Speed
I_{CC} (max): 180 mA
t_{PD} (max): 25 ns

B* = High Speed
in Vorbereitung

–2 = Low Power
I_{CC} (max): 0 = 45 mA, 1 = 90 mA
t_{PD} (max): 0 = 60 ns, 1 = 50 ns

–4 = Low Power
I_{CC} (max): 50 mA
t_{PD} (max): 80 ns

CMOS* und ECL*
in Vorbereitung



Einführung

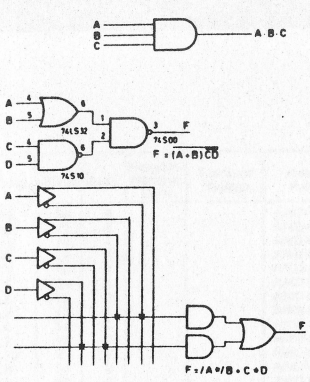
Die PAL-Familie von Monolithic Memories ist ein weiterer Schritt zur Minimierung logischer Schaltungen unter Berücksichtigung einer umfassenden und einfachen Programmierung auf Standard-PROM- oder preiswerten PAL-Programmern.
Das logische Konzept eines PAL-Bausteins ist durch die programmierbare Eingangs-UND-Matrix bei fester ODER-Verknüpfung der Ausgänge gekennzeichnet.

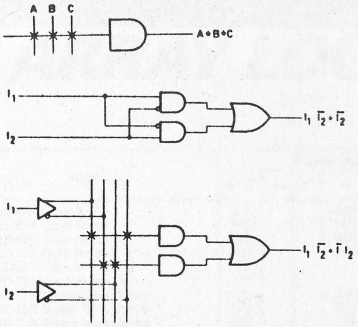
Mit dieser Konfiguration können zunächst sämtliche Grundgatter wie AND, NAND, OR, NOR, INVERTER, EX-OR und EX-NOR realisiert werden. Weiterhin erlaubt dieses Konzept dem Anwender, die Ein- und Ausgänge des Bausteins beliebig festzulegen und damit die Entflechtung der Leiterbahnen bereits auf dem Silizium-Chip durchzuführen.
Die PAL-Familie kann die komplette 74 SSI/MSI-Familie in der Funktion ersetzen.
Die Gatterkomplexität reicht derzeit bis über 500 Gatterfunktionen, Komplexitäten bis zu 2.000 Gattern werden schon in nächster Zukunft verfügbar sein. Eine Verdichtung von Logikbausteinen wird mit mindestens 5:1 erreicht, kann aber je nach Applikation auch wesentlich höher liegen (bis weit über 20:1).
Um jeder Anforderung gerecht zu werden, d.h. Flexibilität mit größtmöglicher Integration von logischen Funktionen zu verbinden, wurden mehrere Familien von PAL-Bausteinen entwickelt, die diese Bedingungen erfüllen (siehe Übersicht auf Seite 3). Dadurch kann der Anwender den Baustein auswählen, der für seine Applikation am besten geeignet ist.
Ein programmiertes PAL kann immer wieder in ein Programmiergerät eingelesen werden, folglich für die Duplizierung weiterer Bausteine dienen. Um sich aber vor unerwünschter Duplizierung seiner erstellten PAL-Programme zu schützen, bietet sich die Möglichkeit, eine letzte Sicherung im Baustein zu trennen, wodurch das PAL zwar seine volle Funktion beibehält, aber nicht mehr in ein Programmiergerät eingelesen werden kann.
Aufgrund der Massenfertigung von Standard-Bausteinen, die dann für die jeweilige Applikation innerhalb von Sekunden programmiert werden können, entstehen keinerlei Initialkosten. Die Durchlaufzeiten zu den ersten Musterbausteinen betragen oft nur wenige Minuten. Die Programmerstellung hierzu sowie die Programmierung des Bausteins wird im Hause des Anwenders auf bereits bestehenden Einrichtungen durchgeführt.

Monolithic Memories bietet hierzu ein PAL-Assemblerprogramm (PALASM™) an, das auf den gängigen µP-Systemen und Rechnern einfach zum Laufen gebracht werden kann.
Mit zahlreichen Second Source-Lieferanten, die das PAL-Konzept pin-, funktions-, technologie- und programmierkompatibel übernommen haben, bietet Monolithic Memories ein vollständiges System programmierbarer Lösungen.

DIE MERKMALE DER PAL®-FAMILIE

- PAL's reduzieren SSI/MSI-Logikbausteine
- Flexibilität durch Programmierbarkeit
- Programmierung auf Standard-PROM-Programmern
- Frei wählbare Pinbelegung
- Entflechtung von Leiterbahnen auf Silizium
- Einfache Prototypenerstellung
- Leiterbahnreduktion
- Vereinfacht die Produktion/Bestückung
- Verringert Testkosten (Eingangskontrolle im Programmiervorgang)
- Software-Unterstützung mit Assembler (PALASM™)
- Alle PAL's im 20/24 Pin Skinny Dip™-Gehäuse (0,3 inch)
- Platzeinsparung
- Erleichtert die Lagerhaltung
- Verbesserung der Systemzuverlässigkeit
- Letzte Sicherung verhindert Duplizierung
- Zu jedem PAL das HAL (Hard Array Logic)
- Multiple Second Source

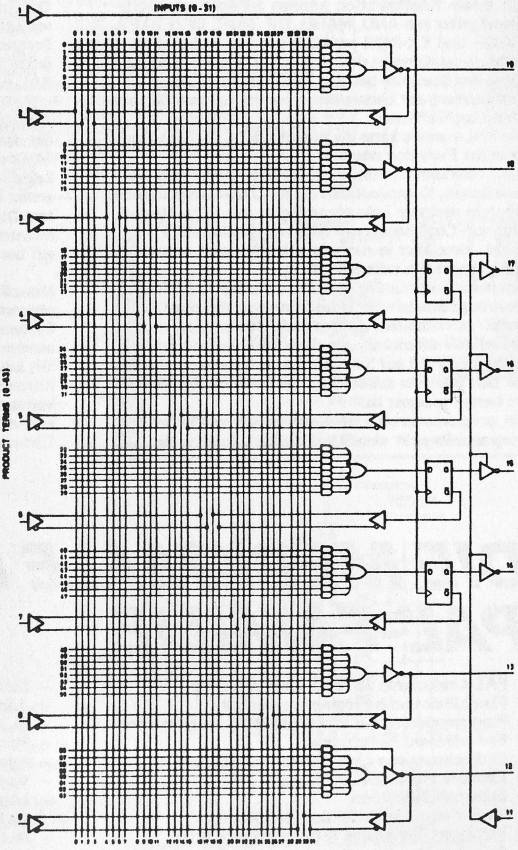




Auf Grund der hohen Integrationsfähigkeit der PAL-Bausteine mußte eine vereinfachte Symbolik der Grundgatter (UND-Funktion) gewählt werden, um das logische Schaltbild eines PAL-Bausteins übersichtlich zu halten.

PAL –SYMBOLIK

LOGIK – DIAGRAMM DES PAL-BAUSTEINS 16R4



PALASM™
PAL-ASSEMBLIERPROGRAMM

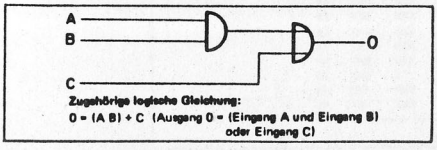
Die gängige Beschreibung von Hardware ist der logische Schaltplan und das Datenblatt. Zum Testen dieser Schaltung wird meist ein Prototyp verwendet. Beide Methoden sind unzulängliche Werkzeuge für die Erstellung von LSI-Baugruppen und zeitkritischen Schaltungen.
Ein logischer Schaltplan mag für menschliche Begriffe verständlich sein, eignet sich aber in den meisten Fällen nicht zur Eingabe in einen Rechner. Auch ein Prototyp stellt einen unbefriedigenden Versuch dar, eine Schaltungsentwicklung zu testen und zu verifizieren. Unterschiede der Durchlaufzeiten, der Schaltungsausführung sowie des Logikkonzepts beeinflussen die Eigenschaften, der mit Schaltplan und Prototyp entwickelten Schaltungen.
Die Kosten eines einzigen während der Entwicklung von hochkomplexen digitalen Schaltungen entstandenen Fehlers sind sehr hoch, außerdem verursachen sie oft monatelange Verzögerungen.
Um eine eindeutige Kommunikation zwischen Anwender und Hersteller integrierter Schaltungen zu gewährleisten, muß eine gemeinsame Sprache zur Definition komplexer LSI-Bausteine vorhanden sein. Das Ziel der Entwicklung einer solchen Sprache sollte sein, unabhängig von Technologie und Hersteller einen digitalen Baustein oder eine entsprechende Schaltung eindeutig beschreiben zu können. Bislang reichten die Diskussionen nur über die Symbolik eines UND-Gatters bis zu der Frage, ob die Normung einer solchen Sprache überhaupt notwendig sei. Diese Uneinigkeit führte aber wiederum zur Entwicklung vieler, nicht kompatibler Termini, die in Logiktestern, CAD-Systemen und für rechnergestützte Schaltungsentflechtung verwendet werden.
Digitale Systeme sind aber eindeutig über *boolesche Zustandsgleichungen* definierbar. Diese Methode erlaubt eine Simulation, ist rechnerunabhängig und sofort auf jedes Computersystem übertragbar.

Entwicklung der Assemblersprache

Ehe logische Gleichungen für die Grundfunktion der Booleschen Algebra aufgestellt werden können, müssen die Operatoren der Sprache (PALASM) festgelegt werden.

= Gleichheitszeichen
:= Gleichheitszeichen für Registerausgänge
IF Bedingung (Three-State-Ausgänge)
/ Komplement
* UND
+ ODER
:+: Exklusiv-ODER
:*: Exklusiv-NICHT-ODER
( ) Klammerzeichen

Die mit Hilfe der Booleschen Algebra formulierten logischen Gleichungen sind Ausgangspunkt für die Programmierung des PAL-Bausteins mittels PALASM™.
Die Eingabe des PAL-Typs, der gewählten Anschlußbelegung und der logischen Gleichungen laut Boolescher Algebra werden entsprechend dem Flußdiagramm in (für das Programmiergerät) geeignete Formate wie Hexadezimal, Binär, BHLF, BPNF etc. übersetzt und können sofort an eine Programmiermaschine überspielt werden.



Simulation und Funktionsprüfung

Die Verdichtung logischer Schaltungen: Die Problematik der Funktionsprüfung digitaler Schaltungen entfernt sich von der Leiterplatte hin zum LSI-Baustein. Mit wachsender Integration logischer Bauelemente werden die Leiterbahnen, die auf einer Leiterplatte SSI-/MSI-Gatter und Flipflops verbinden, auf Silizium übertragen. Die „Entflechtung der Platine" erfolgt bereits auf dem Silizium-Chip. Das bedeutet, daß höher entwickelte Testprogramme vorhanden sein müssen, um ohne Zuhilfenahme interner Testpunkte, wie es noch auf der Leiterplatte möglich war, zu testen.
Die Funktionsprüfung hochintegrierter Bauteile ist kein neues Problem. Standard LSI- und VLSI-Bauelemente wie Mikroprozessoren und deren Peripheriebausteine werden seit einiger Zeit mehr oder weniger erfolgreich getestet. Der Unterschied liegt darin, daß mit programmierbarer Logik in sehr kurzer Zeit viele verschiedene Logikmuster generiert werden können, wobei jedes einzelne Programm ein eigenes Testprogramm benötigt.
Die Vorgehensweise: Testprogramme für LSI-Bausteine bestehen aus einem Parameter- und einem Funktionstest. Die Messung der Parameter wird vom Hersteller vorgenommen; die Funktionsprüfung bleibt dem Anwender überlassen. Dieser logische Funktionstest wird über Testvektoren durchgeführt, die den Anschluß und die logische Bedingung (Low, High, Clock etc.) definieren.
Die Generierung von Testvektoren für einen LSI-Baustein ist aufwendig und zeitraubend, weshalb die Erstellung einem Rechner übertragen wird. Diese Testprogramme benötigen Grundvektoren, die der Entwickler der jeweiligen Schaltung selbst zur Verfügung stellen muß.
Erstellung von Testvektoren mit PALASM: Eine einfache Möglichkeit zur Erstellung von Testvektoren bietet das Simulationsprogramm für PAL-Bausteine von Monolithic Memories.
Der Entwickler erstellt eine einfache Funktionstabelle, aus der die Software Testbedingungen ableitet, die sofort für die Überprüfung der logischen Funktion des PAL-Bausteins verwendet werden können. Das PAL-Simulationsprogramm führt zwei Prüfungen durch:

1. Mit Hilfe der Funktionstabelle werden die logischen Gleichungen auf ihre Richtigkeit geprüft.
2. Die aus dieser Funktionstabelle während der Simulation erzeugten Testvektoren können sofort für die logische Prüfung des Bausteins herangezogen werden.

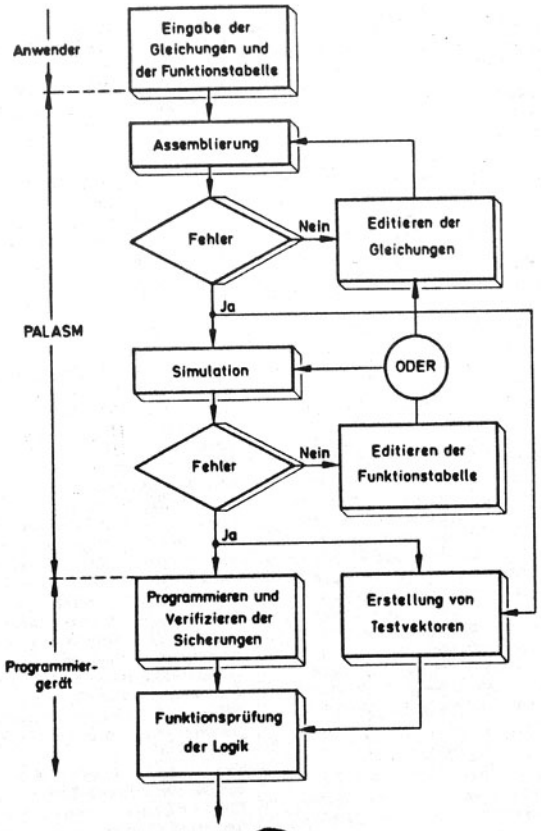
Bei sequentieller Logik können ganze Ablaufdiagramme mit Hilfe dieses Programms simuliert bzw. nach der Programmierung des PAL-Bausteins getestet werden.
Hardware-Realisierung: Die aus der Funktionstabelle des PAL-Programms erzeugten Testvektoren werden mit der Programmierinformation in den „Programmer" geladen und ermöglichen ohne weiteren Aufwand einen 100 %-Logiktest von PALs. Einige der heute am Markt erhältlichen Programmiergeräte bieten bereits diese Möglichkeit an.

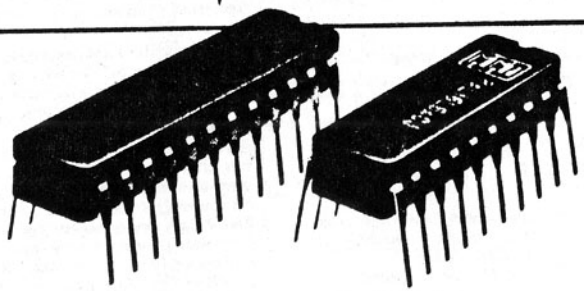
Funktionstabelle

A B C D	Daraus erzeugte Testvektoren
L L L L	1 000XXXXXXOXXXXXXXL1
H L L L	2 100XXXXXXOXXXXXXXL1
L H L L	3 010XXXXXXOXXXXXXXL1
H H L H	4 110XXXXXXOXXXXXXXH1
L L H H	5 001XXXXXXOXXXXXXXH1
H L H H	6 101XXXXXXOXXXXXXXH1
L H H H	7 011XXXXXXOXXXXXXXH1
H H H H	8 111XXXXXXOXXXXXXXH1

Simulation

ABLAUFDIAGRAMM

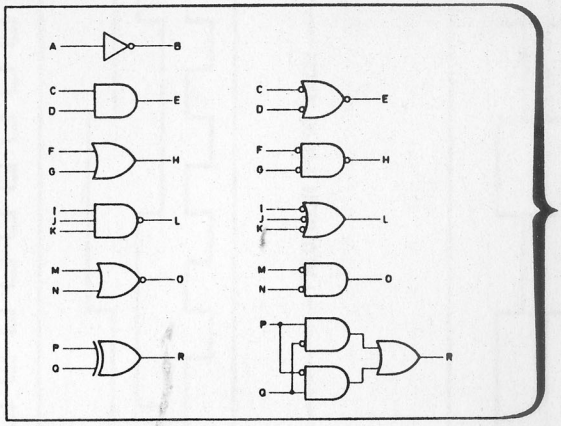


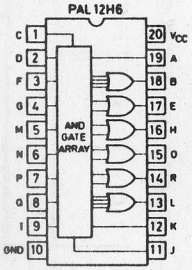


BEISPIEL 1

Beschreibung .

Dieses Beispiel zeigt, wie einfach die grundlegenden Gatterfunktionen wie INVERTER, UND, ODER, NICHT-UND, NICHT-ODER, EXKLUSIV-ODER in einem PAL-Baustein realisiert werden können.





PAL 12H6
PD7000
GATTERFUNKTIONEN
MMI GMBH MÜNCHEN

PAL DESIGN SPECIFICATIONS
WILLY VOLDAN

C D F G M N P Q I GND
J K L R O H E B A VCC

B = /A ; INVERTER
E = C*D ; UND – GATTER
H = F + G ; ODER – GATTER
L = /I + /J + /K ; NICHT – UND – GATTER
O = /M*/N ; NICHT – ODER – GATTER
R = P*/Q + /P*Q ; EXKLUSIV – ODER GATTER

Funktionstabelle

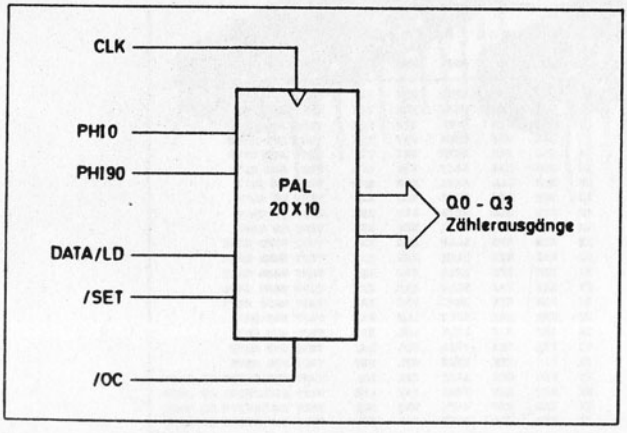
```
A B C D E F G H I J K L M N O P Q R

;AB   CDE   FGH   IJKL   MNO   PQR   COMMENTS
-----------------------------------------------------------------
LH    XXX   XXX   XXXX   XXX   XXX   TEST INVERTER
HL    XXX   XXX   XXXX   XXX   XXX   TEST INVERTER
XX    LLL   XXX   XXXX   XXX   XXX   TEST AND GATE
XX    LHL   XXX   XXXX   XXX   XXX   TEST AND GATE
XX    HLL   XXX   XXXX   XXX   XXX   TEST AND GATE
XX    HHH   XXX   XXXX   XXX   XXX   TEST AND GATE
XX    XXX   LLL   XXXX   XXX   XXX   TEST OR GATE
XX    XXX   LHH   XXXX   XXX   XXX   TEST OR GATE
XX    XXX   HLH   XXXX   XXX   XXX   TEST OR GATE
XX    XXX   HHH   XXXX   XXX   XXX   TEST OR GATE
XX    XXX   XXX   LLLH   XXX   XXX   TEST NAND GATE
XX    XXX   XXX   LLHH   XXX   XXX   TEST NAND GATE
XX    XXX   XXX   LHLH   XXX   XXX   TEST NAND GATE
XX    XXX   XXX   HLLH   XXX   XXX   TEST NAND GATE
XX    XXX   XXX   HHHL   XXX   XXX   TEST NAND GATE
XX    XXX   XXX   XXXX   LLH   XXX   TEST NOR GATE
XX    XXX   XXX   XXXX   LHL   XXX   TEST NOR GATE
XX    XXX   XXX   XXXX   HLL   XXX   TEST NOR GATE
XX    XXX   XXX   XXXX   HHL   XXX   TEST NOR GATE
XX    XXX   XXX   XXXX   XXX   LLL   TEST EXCLUSIVE OR GATE
XX    XXX   XXX   XXXX   XXX   LHH   TEST EXCLUSIVE OR GATE
XX    XXX   XXX   XXXX   XXX   HLH   TEST EXCLUSIVE OR GATE
XX    XXX   XXX   XXXX   XXX   HHL   TEST EXCLUSIVE OR GATE
-----------------------------------------------------------------
```

BEISPIEL 2

Das nachfolgende Beispiel soll demonstrieren, wie aus der Problemstellung in Form eines Ablaufdiagramms über die vier Schritte – Definieren, Simulieren, Produkt, Test – ein PAL-Baustein in Minuten hergestellt wird.

Aufgabenstellung:

Nachfolgendes Impulsdiagramm beschreibt den zeitlichen logischen Ablauf einer Schaltung zur Drehrichtungserkennung. Je nachdem ob das Eingangssignal PHI0 dem Eingangssignal PHI90 vor- oder nachläuft, soll ein nachgeschalteter Binärzähler vorwärts- bzw. rückwärts zählen. Am Zählerstand ist die jeweilige Position der sich bewegenden Teile in binärer Form vorhanden. Der Zähler soll auch gesetzt und rückgesetzt werden können.
Diese Problemstellung ist in einem PAL-Baustein zu lösen.



DEFINIEREN

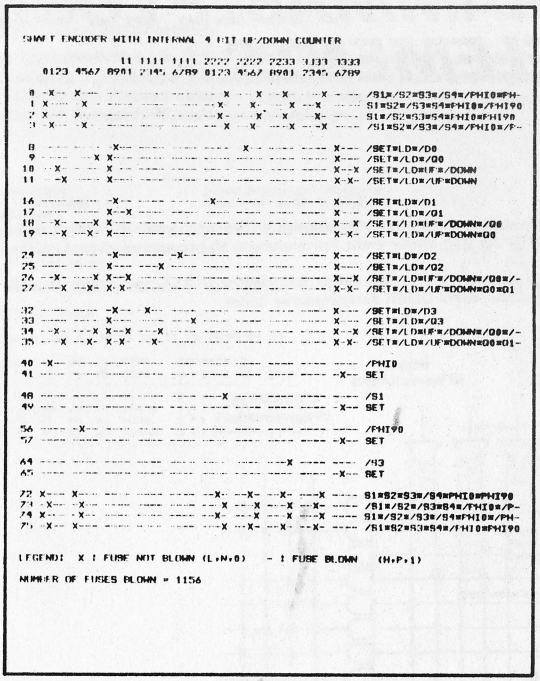
Die Problemstellung (Ablaufdiagramm, Wertetabelle etc.) wird mit Hilfe von logischen Gleichungen laut Boole'scher Algebra definiert.

```
PAL 20X10                          PAL DESIGN SPECIFICATIONS
PD  1510                           WILLY VOLDAN      1/20/87
SHAFT ENCODER WITH INTERNAL 4 BIT UP/DOWN COUNTER
MMI CMOS MUNICH
CLK PHI0 PHI90 /LD D3 D2 D1 D0 NC NC /SET GND
/OC DOWN    S4  S3 S2 S1 Q3 Q2 Q1 Q0  UP  VCC

/S1   :=  /PHI0                              ;CHECK FOR PHI0
      +   SET                                ;INITIALIZE S1=1

/S2   :=  /S1                                ;CHECK FOR S1
      +   SET                                ;INITIALIZE S2=1

/S3   :=  /PHI90                             ;CHECK FOR PHI90
      +   SET                                ;INITIALIZE S3=1

/S4   :=  /S3                                ;CHECK FOR S3
      +   SET                                ;INITIALIZE S4=1

/DOWN :=   S1* S2* S3*/S4*  PHI0* PHI90      ;PHI0 LEADS PHI90  COUNT-FREQ/2
      +  /S1*/S2*/S3*  S4*/PHI0*/PHI90       ;PHI0 LEADS PHI90  COUNT-FREQ/2
      :+:  S1*/S2*/S3*/S4*  PHI0*/PHI90      ;PHI0 LEADS PHI90  COUNT-FREQ/2
      +  /S1*  S2*  S3*  S4*/PHI0*  PHI90    ;PHI0 LEADS PHI90  COUNT-FREQ/2

/UP   :=  /S1*/S2*  S3*/S4*/PHI0*  PHI90     ;PHI90 LEADS PHI0  COUNT-FREQ/2
      +   S1*  S2*/S3*  S4*  PHI0*/PHI90     ;PHI90 LEADS PHI0  COUNT-FREQ/2
      :+:  S1*/S2*  S3*  S4*  PHI0*  PHI90    ;PHI90 LEADS PHI0  COUNT-FREQ/2
      +  /S1*  S2*/S3*/S4*/PHI0*/PHI90       ;PHI90 LEADS PHI0  COUNT-FREQ/2

/D0   :=  /SET* LD*/D0                       ;LOAD D0 (LSB)
      +   /SET*/LD*/D0                       ;HOLD D0
      :+:  /SET*/LD*  UP*/DOWN               ;DECREMENT
      +   /SET*/LD*/UP*  DOWN                ;INCREMENT

/D1   :=  /SET* LD*/D1                       ;LOAD D1
      +   /SET*/LD*/D1                       ;HOLD D1
      :+:  /SET*/LD*  UP*/DOWN*/D0           ;DECREMENT
      +   /SET*/LD*/UP*  DOWN*  D0           ;INCREMENT

/D2   :=  /SET* LD*/D2                       ;LOAD D2
      +   /SET*/LD*/D2                       ;HOLD D2
      :+:  /SET*/LD*  UP*/DOWN*/D0*/D1       ;DECREMENT
      +   /SET*/LD*/UP*  DOWN*  D0*  D1       ;INCREMENT

/D3   :=  /SET* LD*/D3                       ;LOAD D3 (MSB)
      +   /SET*/LD*/D3                       ;HOLD D3
      :+:  /SET*/LD*  UP*/DOWN*/D0*/D1*/D2   ;DECREMENT
      +   /SET*/LD*/UP*  DOWN*  D0*  D1*  D2   ;INCREMENT
```

SIMULIEREN

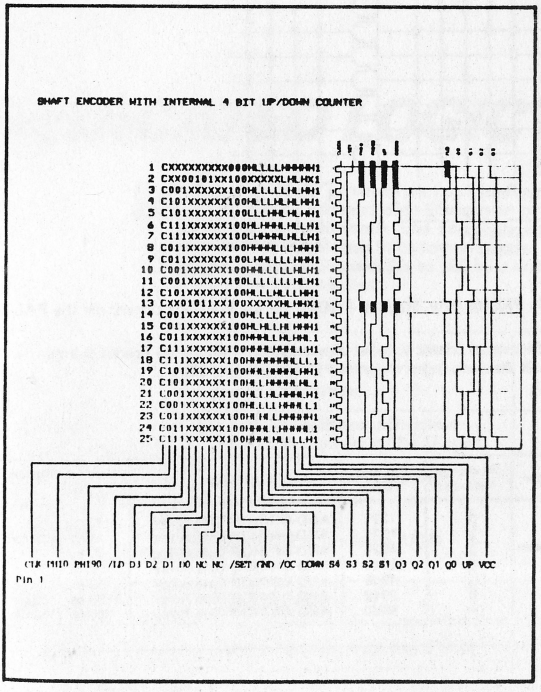
Über eine Funktionstabelle überprüft das PAL-Simulationsprogramm Zeile für Zeile, ob die vom Entwickler erstellten Gleichungen hinsichtlich der Logik und dem Zeitverhalten der Problemstellung entsprechen.

```
FUNCTION TABLE

/OC CLK /SET /LD  PHI0 PHI90 S1 S2 S3 S4 UP DOWN D3 D2 D1 D0 Q3 Q2 Q1 Q0

|-CONTROL-  <INPUTS>                    -DATA-  -DATA-
|           P                             IN      OUT
|   /       P H                   D
| /C C S /  H I                   O
|O L E L    I 9    8888  U  M     DDDD    QQQQ
|C K T D    8 8    1234  P  N     3210    3210   COMMENTS (HEX)
-------------------------------------------------------------------------
 L C L X    X X    LLLL  H  H     XXXX    HHHH   INITIALIZE REGISTERS
 L C H L    X X    XXXX  X  X     LHLH    LHLH   LOAD (5)
 L C H H    L L    LLLL  H  H     XXXX    LHLH   HOLD (5)
 L C H H    H L    HLLL  H  H     XXXX    LHLH   HOLD (5) PHI0 LEADS PHI90
 L C H H    H L    HHLL  H  L     XXXX    LHLH   HOLD (5)
 L C H H    H H    HHHL  H  H     XXXX    LHLL   DECREMENT (4)
 L C H H    H H    HHHH  H  L     XXXX    LHLL   HOLD (4)
 L C H H    L H    LHHH  H  H     XXXX    LLHH   DECREMENT (3)
 L C H H    L H    LLHH  H  L     XXXX    LLHH   HOLD (3)
 L C H H    L L    LLLH  H  H     XXXX    LLHL   DECREMENT (2)
 L C H H    L L    LLLL  H  L     XXXX    LLHL   HOLD (2)
 L C H H    H L    HLLL  H  H     XXXX    LLLH   DECREMENT (1)
 L C H L    X X    XXXX  X  X     HLHH    HLHH   LOAD (B)
 L C H H    L L    LLLL  H  H     XXXX    HLHH   HOLD (B)
 L C H H    L H    LLHL  H  H     XXXX    HLHH   HOLD (B) PHI0 LAGS PHI90
 L C H H    L H    LLHH  L  H     XXXX    HLHH   HOLD (B)
 L C H H    H H    HLHH  H  H     XXXX    HHLL   INCREMENT (C)
 L C H H    H H    HHHH  L  H     XXXX    HHLL   HOLD (C)
 L C H H    H L    HHLH  H  H     XXXX    HHLH   INCREMENT (D)
 L C H H    H L    HHLL  L  H     XXXX    HHLH   HOLD (D)
 L C H H    L L    LHLL  H  H     XXXX    HHHL   INCREMENT (E)
 L C H H    L L    LLLL  L  H     XXXX    HHHL   HOLD (E)
 L C H H    L H    LLHL  H  H     XXXX    HHHH   INCREMENT (F)
 L C H H    L H    LLHH  L  H     XXXX    HHHH   HOLD (F)
 L C H H    H H    HLHH  H  H     XXXX    LLLL   INCREMENT (0) ROLL OVER
-------------------------------------------------------------------------
```



PRODUKT

Das PAL-Assemblierprogramm PALASM übersetzt die logischen Gleichungen in verschiedene, für Programmiergeräte geeignete Datenformate (Binär, Hexadezimal, BHLF, BPNF etc.) um die Bausteine zu programmieren.
Das PALASM legt damit die Geographie des Bausteins fest.

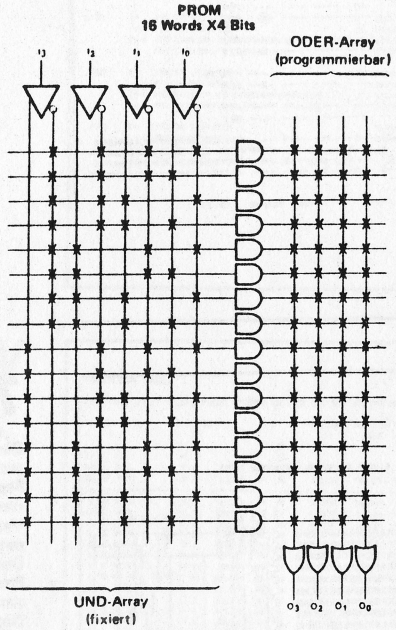


TEST

Die aus dem Simulationsvorgang erzeugten Testvektoren werden ebenfalls vom Programmer übernommen um sofort nach dem Programmiervorgang eine logische Verifizierung des Bausteins durchzuführen.
Der Programmer überprüft damit, ob die Problemstellung im PAL-Baustein realisiert ist.
Der Entwicklungs- und Testvorgang ist abgeschlossen.

PLE™ PROGRAMMABLE LOGIC ELEMENTS

Wurden bipolare PROMs über die letzten Jahre vorwiegend als Speicher eingesetzt (Mikroprogrammspeicher, Look-up Tables, Charaktergeneratoren usw.), so finden sie zur Zeit immer mehr Anwendungen als programmierbare Logikbausteine.
Die PLE (PROM)-Konfiguration besteht im Gegensatz zu PAL aus einem festgelegten UND-Eingangsarray, das die Adressdekodierung für das nachfolgende programmierbare ODER-Ausgangsarray bildet.



Monolithic Memories entwickelte ultraschnelle PROMs bzw. Register-PROMs für Logikanwendungen, die die PAL-Familie ergänzen.

Der Einsatz von PLEs (Programmable Logic Elements) bietet sich bei Anwendungen mit vielen Produkttermen (bis 4096), PAL (Programmable Array Logic) für Anwendungen mit vielen Eingängen (bis 64) an.

PLE™ Programmable Logic Elements

Bezeichnung	Eing.	Ausg.	Organisation	Register	Ausgangs-Polarität	Produkt-Terme	Funktion	Geschwindigkeit*	
								STD	A
63S081	5	8	32 x 8	asynchron	programmierbar	32	AND-OR/NOR Gate Array	25 ns	
63S441	10	4	1K x 4			1024	AND-OR/NOR Gate Array	45 ns	35 ns
63RA441	10	4	1K x 4			1024	AND-OR/NOR G.A. w/Reg's	30 ns	
63S841	11	4	2K x 4			2048	AND-OR/NOR Gate Array	50 ns	35 ns
63S1641	12	4	4K x 4			4096	AND-OR/NOR Gate Array	50 ns	35 ns
63S1681	11	8	2K x 8			2048	AND-OR/NOR Gate Array	50 ns	35 ns
63S3281	12	8	4K x 8			4096	AND-OR/NOR Gate Array	50 ns	40 ns

* Alle Angaben sind Max.-Werte

PLEASM™
PLE-Assemblierprogramm

Die Logikentwicklung in PLE (PROM) wird mit dem PLE-Assembler PLEASM™ von Monolithic Memories unterstützt. Ähnlich dem PALASM werden mit Hilfe Boole'scher Algebra logische oder arithmetische Funktionen in verschiedene Formate übersetzt (Binär, Hexadezimal, BHLF, BPNF etc.) um dann das Bauelement sofort auf Standard-PROM-Programmiergeräten zu programmieren.

```
PROM1024X4                          PROM DESIGN SPECIFICATION
PATTERN NUMBER 2                    WILLY VOLDAN  08/22/82
ADDIERER
MMI GMBH MUENCHEN

.ADD A0 A1 B0 B1 C0 C1 D0 D1 E0 E1
.DAT P0 P1 P2 P3

P3,P2,P1,P0 = A1,A0 .*. B1,B0 .*. C1,C0 .*. D1,D0 .*. E1,E0

FUNCTION TABLE

A1 A0 B1 B0 C1 C0 D1 D0 E1 E0 P3 P2 P1 P0

.AA BB CC DD EE PPPP     COMMENTS
:10 10 10 10 10 3210     A + B + C + D + E = P
-- -- -- -- -- ----     -------------------
LL LL LL LL LL LLLL     0 + 0 + 0 + 0 + 0 =  0
LH LH LH LH LH LHLH     1 + 1 + 1 + 1 + 1 =  5
HL HL HL HL HL HLHL     2 + 2 + 2 + 2 + 2 = 10
HH HH HH HH HH HHHH     3 + 3 + 3 + 3 + 3 = 15
```

PROM ASSEMBLER OPERATORS
(IN HIERARCHY OF EVALUATION)

;	COMMENT FOLLOWS
.	DOT OPERATOR (KEY OPERATOR OR ARITHMETIC FOLLOWS)
ADD	ADDRESS PINS (INPUTS)
DAT	DATA PINS (OUTPUT)
,	DELIMINATOR, SEPARATES BINARY BITS (MSB FIRST)
=	EQUALITY (COMBINATORIAL)
:=	REPLACED BY AFTER THE CLOCK (SEQUENTIAL)

BOOLEAN OPERATORS

/	COMPLEMENT, PREFIX TO A PIN NAME
*	AND (PRODUCT)
+	OR (SUM)
:+:	XOR (EXCLUSIVE OR)
:*:	XNOR (EXCLUSIVE NOR)

ARITHMETIC OPERATORS

.*.	MULTIPLY (ARITHMETIC MULTIPLICATION)
./.	DIVIDE (ARITHMETIC DIVISION)
.+.	PLUS (ARITHMETIC ADDITION)
.-.	MINUS (ARITHMETIC SUBTRACTION)

USER DEFINED OPERATORS

.SIN.	TRIG SININE
.COS.	TRIG COSINE
.EXP.	EXPONENTIAL
.LN.	NATURAL LOG

Operanden für die logischen Gleichungen der Boole'schen Algebra

DEDICATED LOGIC

Unter "Dedicated Logic" verstehen wir Logikkonfigurationen, die sich, bedingt durch ihre parallele Datenstruktur und aus kommerziellen Erwägungen nicht zu weiterer Integration eignen.
Die Bauelemente sind Arithmetik- und Logik-ICs, deren Eigenschaften aus Geschwindigkeits- und Kostengründen mit festgelegten Strukturen am besten realisiert werden können.
Diese Bausteine sind für Systeme (bis zu 100 MHz und niedrigsten Verlustleistungen in CMOS) mit festgelegter Funktion konzipiert.

Bezeichnung	Funktion	Eigenschaft
SN74S557 SN74S558	8 x 8 Multiplizierer	60 ns max. Multiplikationszeit
SN74S508 SN74S516	8 x 8 Multiplizierer / Dividierer 16 x 16 Multiplizierer / Dividierer	0,8 µs Multiplikation, 2,2 µs Division 1,5 µs Multiplikation, 3,5 µs Division
SN74S408 SN74S409	Dynamischer RAM-Controller	Steuert bis zu 88 D-RAMs und/oder 1 Megawort
SN74S700 SN74S730 SN74S731 SN74S734	Treiber für dynamische RAMs	Steuert bis zu 500 pF-Last Integrierter serieller Widerstand
SN74LS Series SN74S Series	Octal-Interface	Register, Latches, Transceiver, Buffer
C67401 C67402	First-In-First-Out-Speicher (FIFO)	10, 15, 16,7 25 MHz Geschwindigkeit

HAL® HARD ARRAY LOGIC

Um die programmierbaren Lösungen zu unterstützen, bietet Monolithic Memories die Bausteine der gesamten PAL-Familie auch als HAL (Hard Array Logic) an.
HALs werden vom Hersteller maskenprogrammiert und ermöglichen dem Kunden weiter reduzierte Stromaufnahme sowie kosteneffektiveren Einsatz in der Produktion. Der Vorteil ist eine sofortige Prototypenerstellung in PAL mit Dokumentation mittels PALASM um dann ohne Wartezeiten die Serienfertigung kostengünstiger mit HAL-Bausteinen durchzuführen.

VERGLEICH VON LOGIK-KONZEPTEN

Logik-Konzepte	Logik-Ebene			Transistor-Ebene		Bemerkungen
	Std. SSI/MSI TTL/CMOS	PAL HAL ECL TTL/CMOS	Gate Array ECL/TTL/CMOS		Full Custom	
Entwicklungs-kosten	1	0,1	10	50	100	Aequivalent einer Europa-Karte mit 1000 Gatterfunktionen
Entwicklungs-zeit	1	0,1	10	50	100	Zeit = Kosten
Entwicklungs-hilfe	Datenbuch Lötkolben 1	Programm-mer 5	CAD Großrechn. 1000	Konzept spezifische CAD/CAM Rechner 5000		Normiert auf 1 = DM 1000,00
Flexibilität	1	10	0,1	0,1	0,01	Durch Programmierung von TiW-Sicherungen, Konzeptänderungen und Kundenadaptionen möglich
Entwicklungs-risiko	1	<1	>1	>	>	„First time success"
Prototyp-Erstellung	2 Tage	2 Stunden	10 Wochen + 16 Wochen	20 Wochen + 16 Wochen	1 Jahr	Feld programmierbar/ Masken-Erstellung
Kosten pro Gatter	DM 0,10	DM 0,03	0,02 –	– 0,05	0,015	Ohne Entwicklungskosten für Maskenerstellung und Testvektorgenerierung
Gatter-Komplexität	SSI MSI	50 – 2000 Progr.	50 – 5000		Bis 10 000	Ab 64 Macrozellen zu 32 Gatter/Zelle ist die allumfassende Verknüpfbarkeit problematisch = AEQU. 2048 Gates
Dokumentation (Service)	Gut	Sehr gut	Proble-matisch	Sehr proble-matisch	Sehr proble-matisch	Entsteht während der Entwicklung durch PALASM für Produktion, QA, QC und Service
MMI +	Die Second Source					

Stand 1 Quartal 1983

BEZEICHNUNGSSYSTEM BESTELLINFORMATION

PAL

PAL 16 L 8 A – 2 M J 883 B

PAL
Programmable Array Logic
HAL
Hard Array Logic

Anzahl der Eingänge

Ausgangstyp
L = Active Low
H = Active High
P = programmierbare Ausgangspolarität
X = Exklusiv-ODER-Register
A = Arithmetik-Register
C = Komplementärausgang
R = Register
FF = Flip-Flops mit asynchr. Clear, Clock u. Preset
RP = Register mit progr. Ausgangspolarität

Anzahl der Ausgänge

wahlweise Hi-Rel-Prozesse
883 B = Mil-Std.-883, Methode 5004 & 5005 Klasse B
SHRP = MMI-eigenes Hi-Rel-Programm mit Burn-in

Gehäuse
J = Keramik DIP
N = Plastik DIP
F = Flachgehäuse (Flat Pack)
L = Leadless Chip Carrier (LCC)

Temperaturbereich
M = mil. (–55 °C bis +125 °C)
C = kom. (0 °C bis +75 °C)

Stromaufnahme
–2 = c.50% von Standard
–4 = c.25% von Standard

Geschwindigkeit
A = High Speed 15 ns typ.
B = High Speed 10 ns typ.

PLE

6 3 S 32 8 1 A

Temperaturbereich
6 = kommerziell
5 = militärisch

Produkt
3 = PROM

Familie
S = Schottky
LS = Low Power Schottky
RX = Register

A = höhere Geschwindigkeit

Ausgangstyp
0 = Open Collector
1 = Three State

Anzahl der Ausgänge
4 = 4 Bit
8 = 8 Bit

Speichergröße
0 = 256 Bit
1 = 1024 Bit
2 = 2048 Bit
4 = 4096 Bit
8 = 8192 Bit
16 = 16384 Bit
32 = 32768 Bit

Die Bezeichnungssysteme der weiteren Bauteil-Familien von Monolithic Memories wie Arithmetik-Elemente, Octal-Interface, FIFO, D-RAM-Controller etc. entnehmen Sie bitte aus dem speziellen Informationsmaterial, das wir Ihnen auf Wunsch gerne zusenden (siehe Antwortkarte Umschlag).

NUCLEAR INTERFACE
Goldstraße 64
D-4400 Münster
Telefon: 02 51-27 35 85