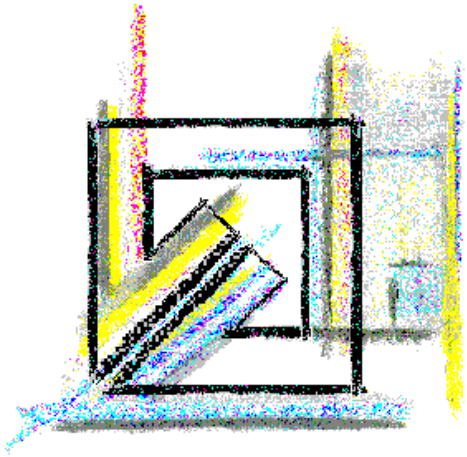
Fachhochschule
Braunschweig/Wolfenbüttel

Labor für Datentechnik

Prof. Dr.-Ing. R. Bermbach



Versuch: PLD - Programmierbare Logikbausteine

Inhaltsverzeichnis

1 Einleitung

Mit dem Aufkommen der Digitaltechnik und der sich sprunghaft entwickelnden Mikroprozessor-Technik in den 70er Jahren stieg der Wunsch nach programmierbarer Logik. Durch sie wird die Möglichkeit geschaffen, ein ganzes „TTL-Grab“ durch ein einziges PLD zu ersetzen. PLDs (Programmable Logic Devices) sind ICs, die erst vom Anwender durch Programmierung in ihrer Funktion festgelegt werden. Erhöhte Flexibilität, Platz- und Kostenersparnis, sowie Vorteile in der Entwicklungsphase verwirklichten bereits die ersten programmierbaren Logikbausteine (PLDs) - die PROMs. Es folgten die PALs und andere Logikbausteine aus der Gruppe der PLDs, bis die Firma LATTICE Mitte der 80er Jahre den ersten wiederprogrammierbaren Logikbaustein, das GAL16V8, vorstellte. Bis zu diesem Zeitpunkt wurden falsch programmierte Bausteine entweder weggeworfen (PROMs und PALs) bzw. sie mußten zum Löschen einige Minuten lang mit UV-Licht bestrahlt werden.Von da an entwickelte sich der PLD-Sektor rasant. Die Logikkapazitäten der PLDs stiegen, Stromaufnahme und Signallaufzeiten wurden drastisch gesenkt. Den wiederprogrammierbaren Logikbausteinen folgten die „im System“ programmierbaren Logikbausteine. Diese Bausteine können direkt im Zielsystem (in der Anwendungsschaltung) programmiert werden. Ein spezielles - meistens teures - Programmiergerät ist zum Programmieren dieser Bausteine nicht mehr notwendig und die Herausnahme des Bausteins aus dem Zielsystem für den Programmiervorgang entfällt hierbei.

Neben der Hardware wurde auch die zum Logikentwurf notwendige Software ständig weiterentwickelt. Wurden die ersten PLDs noch „per Hand“ programmiert, so gibt es mittlerweile ein fast unüberschaubares Angebot an entsprechender Software, die dem Entwickler viel Arbeit abnehmen kann und schon im Vorfeld - z. B. durch Simulation - Fehler verhindern hilft. Bestimmte Aufgaben, wie z.B. die Entwicklung von Boolschen Funktionsgleichungen aus Wahrheitstabellen, werden der Software übertragen und der Entwickler kann sich dadurch ganz auf den Gesamtentwurf konzentrieren. Hardwarebeschreibungssprachen (HDL - Hardware Description Language) helfen komplexe logische Zusammenhänge einfach zu beschreiben. Die „Beschreibung“ eines Logikmodells bzw. Logikentwurfs kann auch grafisch erfolgen, in diesem Fall durch Funktionsblöcke, deren Funktion durch eine HDL beschrieben ist.

Zu dem Versuchsplatz gehört ein PC, auf dem u.a. ein Programm installiert ist, das die Versuchsteilnehmer durch den Versuch führt. Um eine interaktive Benutzerführung und Versuchsbegleitung zu ermöglichen, wurde das Programm mit dem interaktiven Autorensystem ToolBook realisiert . Damit wurde es möglich, den Wissensstand der Versuchsteilnehmer während der Versuchsdurchführung zu prüfen und automatisch zu bewerten. Diese Lösung hält den Arbeits- und Betreuungsaufwand für den Laborbetreuer gering und entlastet gleichzeitig die Versuchsteilnehmer, indem die sonst übliche schriftliche Nachbereitung des Versuchs entfällt.

Das Programm teilt sich in mehrere Abschnitte auf. Der Versuchsteilnehmer wird durch diese Abschnitte geführt. Er kann die Reihenfolge der Berarbeitung der einzelnen Abschnitte bestimmen und Abschnitte wiederholen. Im ersten Abschnitt wird die Bedienung des Programms erklärt und der Versuch als Ganzes kurz vorgestellt. Der darauffolgende Abschnitt vermittelt die Grundlagen der PLDs. Neben den Grundlagen wird hier auch speziell auf das GAL20V8 eingegangen, um den Praxisbezug zu gewährleisten und um das notwendige Wissen für den Abschnitt „Praktische Anwendungen“ zu vermitteln.

Im nächsten Abschnitt prüfen Multiple-Choice-Fragen das angeeignete Wissen. Anschließend folgt eine Einführung in die Hardwarebeschreibungssprache ABEL. Verschiedene Methoden der Logikbeschreibung werden vorgestellt; von der einfachen Beschreibung durch Funktionsgleichungen über die Beschreibung durch Wahrheitstabellen bis hin zu Zustandsdiagrammen. Programmieraufgaben, die automatisch auf Richtigkeit überprüft werden, runden diesen Abschnitt ab. Der letzte Versuchsteil beschäftigt sich mit der praktischen Anwendung der neu erlangten Kenntnisse. Hier muß eine Anwendung mit dem GAL20V8 als Kernstück realisiert werden. Basierend auf einer speziellen Aufgabenstellung muß ein GAL20V8 so programmiert werden, daß die Anwendung wie gefordert funktioniert. Die Versuchsteilnehmer können aus dem folgenden Angebot ein für sie interessantes Thema auswählen: Einfache Kodier- und Dekodierlogik (Parallel-A/D-Wandler), PLL-Technik (Frequenzsynthesizer), Dual-Slope-Umsetzer (Voltmeter) und die Steuerung von Gleich- und Stepper-Motoren.

Der Versuchsplatz kann zusätzlich zu seiner eigentlichen Bestimmung von Diplomanden oder Studenten dazu genutzt werden, eigene (GAL-) Anwendungen zu realisieren. Eigens für Interessierte, die privat GALs programmieren möchten, wurde vom Autor ein Programmiergerät entwickelt. Zusammen mit der Programmiersoftware aus diesem Versuch kann so ein sehr effizienter und komfortabler GAL-Entwicklungsplatz geschaffen werden. Weiterhin wurde für den Versuchsplatz u.a. ein Versuchsbrett mit einer Mikrocontroller-Einheit entwickelt, um die praktischen Anwendungen realisieren zu können. Somit können sich Studenten bei bestehenden Interesse mit der Mikrocontroller-Technik auseinandersetzen und erste Erfahrungen auf diesem Gebiet sammeln.

1.1 Versuchsvorbereitung

Dieser Versuch bietet Ihnen die Möglichkeit, sich Wisssen aus dem Gebiet der Programmierbaren Logikbausteine anzueignen. Nutzen Sie diese Chance !

Grundlagen der Digitaltechnik werden für diesen Versuch unbedingt vorausgesetzt. Dazu gehören der Umgang mit Boolscher Algebra und Funktionsgleichungen, sowie Kenntnisse u.a. über das RS-Flipflop und dessen Funktionsgleichung [5]. Die Grundschaltungen der Operationsverstärkertechnik sollten Ihnen geläufig sein. Unterschätzen Sie diesen Versuch nicht aufgrund der einfach erscheinenden Grundlagen ! Erarbeiten Sie sich das **Kapitel 4** (außer 4.4) **besonders gründlich**. Lassen Sie sich nicht von der komplexen Ausgangszelle in Abb. 4.5 abschrecken. **Wichtig ist, daß Ihnen die Unterschiede zwischen den 3 möglichen GAL-Betriebsmodi klar werden**. Zusätzlich zu der vorliegenden Versuchsunterlage kann die Literaturquelle [5] - aber nur die 3. Auflage, die ersten beiden Auflagen stellen die Betriebsmodi fehlerhaft dar - empfohlen werden. Neben den Grundlagen erwartet Sie eine Einführung in ABEL-HDL. Diese Einführung ist der zeitintensivste Teil des ganzen Versuchs. Eine neue Programmiersprache kann immer am besten durch praktische Beispiele erlernt werden. Deshalb umfaßt Kapitel 5 nur ein kurzes Beispiel. Studieren Sie dieses Beispiel um einen ungefähren Eindruck von der Problematik zu bekommen. Zweckmäßig ist das Durcharbeiten von *abel_Anl.htm* (20 seitiger Crash-Curs).

Eine schlechte Vorbereitung führt zu einer Verlängerung der Versuchsdauer ! Gut vorbereitet sind etwa 3,5 Stunden für den kompletten Versuch anzusetzen.

2 Grundlagen der programmierbaren Logik

2.1 Prinzipieller Aufbau der PLDs

In diesem Kapitel wird auf die einzelnen Elemente der PLDs näher eingegangen. Wie aus Abb. 2.1 ersichtlich ist, setzt sich die Struktur einer PLD aus folgenden Elementen zusammen:

- dem Eingabeblock
- der programmierbaren UND-/ODER-Matrix
- der programmierbaren Rückkopplung
- dem Ausgabeblock

Abb. 2.1 Die allgemeine Struktur der PLDs

Nicht alle PLDs müssen die genannten Elemente realisieren können. Kern aller PLDs ist die programmierbare UND-/ODER-Matrix. Da die meisten PLDs eine UND-Matrix besitzen wird ab hier, aus Gründen der Vereinfachung, der Begriff UND-Matrix verwendet. Um einen leichten Einstieg in die Thematik zu ermöglichen, wird im Weiteren eine Beispiel-PLD zur Veranschaulichung herangezogen. Eine reale PLD ist wesentlich komplexer aufgebaut, was an dieser Stelle aber nur Verwirrung stiften würde. Hier folgt nun die erwähnte Beispiel-PLD:

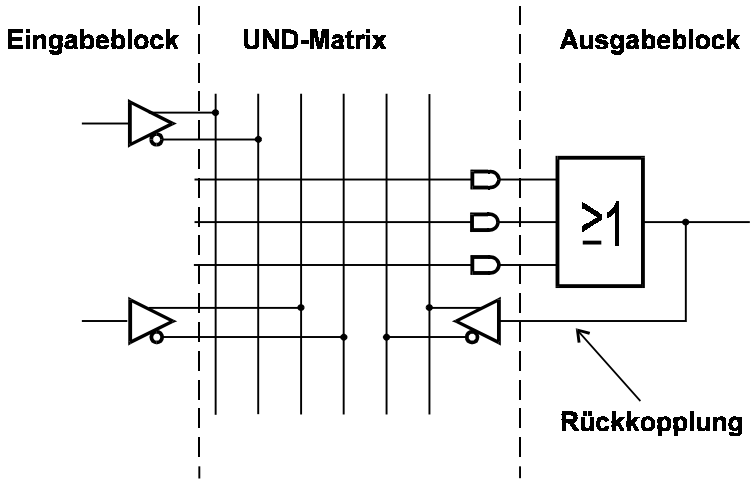


Abb. 2.2 Die Beispiel-PLD

2.1.1 Der Eingabeblock

Der Eingabeblock hat die Aufgabe, die Eingangssignale in die UND-Matrix zu führen. Dabei finden vorwiegend zwei Schaltungsvarianten Verwendung:

- Die Eingangssignale gelangen über Treiber und Inverter in die UND-Matrix. Somit stehen die Eingangssignale sowohl invertiert als auch nicht-invertiert für Verknüpfungen zur Verfügung.

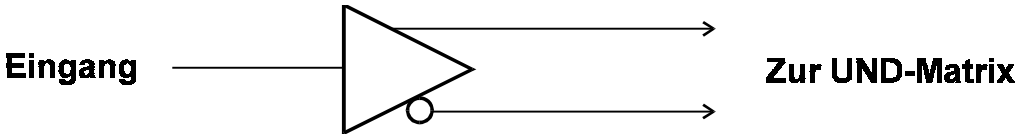


Abb. 2.3 Eingangs-Treiber/-Inverter

- Zwischen den Eingängen und der UND-Matrix sind D-Flipflops geschaltet

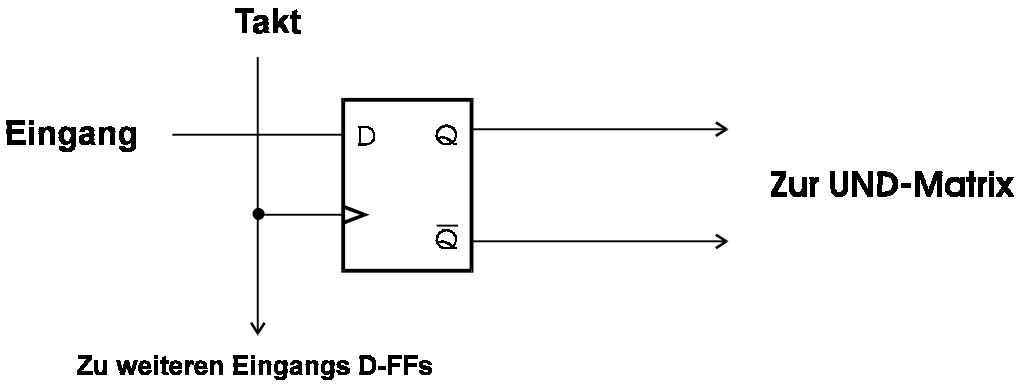


Abb. 2.4 Eingang mit nachgeschaltetem D-FF

2.1.2 Die UND-Matrix

In der UND-Matrix (AND-Array) werden die UND-Verknüpfungen der Produktterme realisiert. Die Matrix ist eine übersichtliche Darstellungsform dieser Verknüpfungen. Die Spalten stellen dabei die Eingangs- und Rückkopplungssignale dar, die innerhalb einer Zeile zu einem Produktterm UND-verknüpft werden. In der Matrix wird das dazu notwendige UND-Gatter der Übersichtlichkeit halber nicht dargestellt. Kreise (oder Kreuze) an den Kreuzungspunkten geben an, welche Signale in einer Zeile miteinander verknüpft werden (Abb. 2.5). An jedem Kreuzungspunkt befindet sich eine „Sicherung“, die sofern sie noch intakt ist, symbolisch durch einen Kreis (oder Kreuz) gekennzeichnet ist. Diese Sicherungen können beim Programmiervorgang gelöscht werden (kein Symbol an der betreffenden Stelle). Nähere Informationen zum Thema programmierbare Elemente („Sicherungen“) befinden sich im Abschnitt 2.2 Programmiertechnologien.

Zur Veranschaulichung dient wieder die Beispiel-PLD:

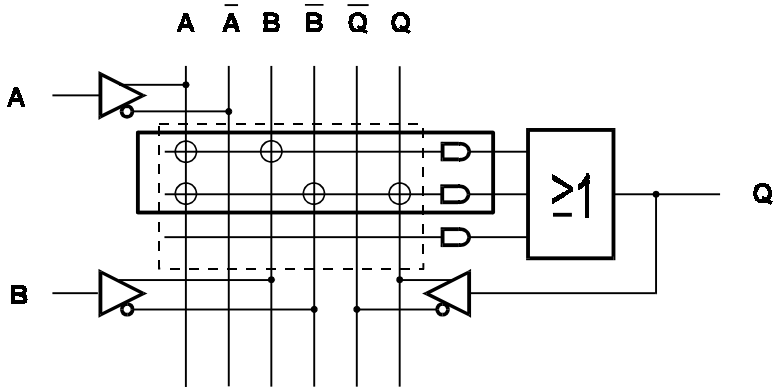


Abb. 2.5 Beispiel-PLD (programmiert)

Programmierbar sind ausschließlich die Kreuzungspunkte im gestrichelt eingerahmten Bereich ! Programmierbare Elemente außerhalb dieses Bereichs ergäben keinen Sinn. Die untere Abbildung zeigt den dick umrahmten Bereich der Oberen im Detail:

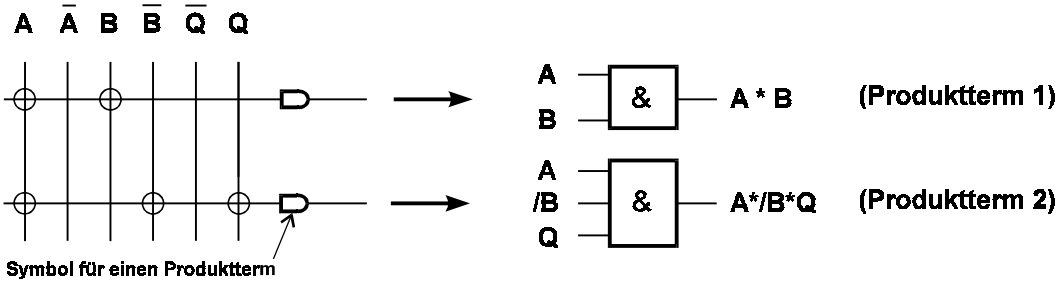


Abb. 2.6 Bildung von zwei Produkttermen

Die Beispiel-PLD läßt sich folgendermaßen charakterisieren:

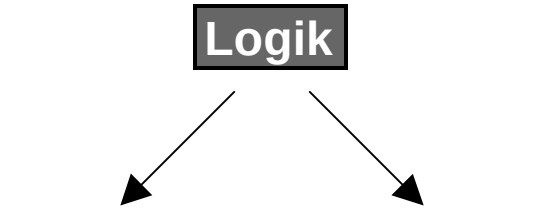
- 6 x 3 UND-Matrix - 6 programmierbare Kreuzungspunkte horizontal und jeweils drei vertikal
- max. 3 Produktterme - 3 Produktterm-Symbole am Ausgangsgatter, daraus ergibt sich die Ausgangs-Gleichung: Q = PT1 + PT2 +PT3

Achtung !!!

In diesem Beispiel sind Produktterme mit bis zu sechs UND-Verknüpfungen möglich, aber nur maximal drei machen einen Sinn. Sind es mehr als drei, so wird mindestens eines der Signale zwangsläufig mit seinem invertierten Zustand UND-verknüpft und man erhält, als Ergebnis für den Produktterm den Wert 0 (A * /A = 0). Ein Produktterm ist in diesem fall inaktiv (=unprogrammiert) und beeinflußt das Ausgangssignal nicht !

2.1.3 Die Rückkopplung

Bei einer Rückkopplung wird das Ausgangssignal in die UND-Matrix zurückgeführt. Die Rückkopplung ermöglicht erst die Einbeziehung des Ausgangssignals in Verknüpfungen. Um die Notwendigkeit von Rückkopplungen zu erkennen, muß man wissen, daß sich Logik in kombinatorische- und sequentielle Elemente aufteilen läßt. Zur Verdeutlichung dient Abb. 2.7:



Kombinatorik	Sequentielle Elemente
Funktionen:	Funktional:
NICHT	D-Flipflop
UND	T-Flipflop
ODER	RS-Flipflop
EXKLUSIV-ODER	JK-Flipflop
ÄQUIVALENZ	LATCH

Abb. 2.7 Die logischen Elementarfunktionen

Ohne die Möglichkeit einer Rückkopplung sind sequentielle Elemente nicht realisierbar. Als Beispiel zeigt die folgende Abbildung ein RS-Flipflop:

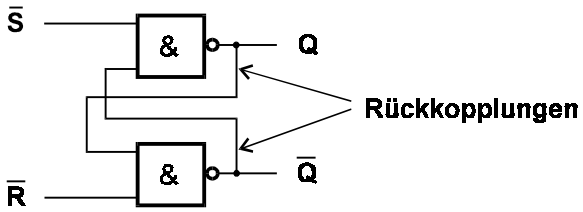


Abb. 2.8 RS-Flipflop mit Rückkopplungen

Sequentielle Elemente können selbstverständlich auch durch entsprechende Verdrahtung außerhalb des Bausteins mit Standardlogik (z.B.74XX) realisiert werden. Hierin besteht nun ein großer Vorteil der PLDs, durch programmierbare Rückkopplungen innerhalb des PLDs entfällt zusätzlicher Verdrahtungsaufwand. Dies trägt zu einer erhöhten Flexibilität bei.

2.1.4 Der Ausgabeblock

Der Ausgabeblock verarbeitet die Ergebnisse der UND-Verknüpfungen (Produktterme) und stellt sie am Ausgang zur Verfügung. Bestandteile eines Ausgabeblocks können sein:

- ODER-Gatter mit mehreren Eingängen (fast immer vorhanden)
- Tri-State Buffer
- EXOR-Gatter zur Signalinvertierung (Ausgangs-Polarität)
- T-, RS-, JK-, D-Flipflops

Abb. 2.9 zeigt ein Beispiel für eine mögliche Konfiguration eines Ausgabeblocks:

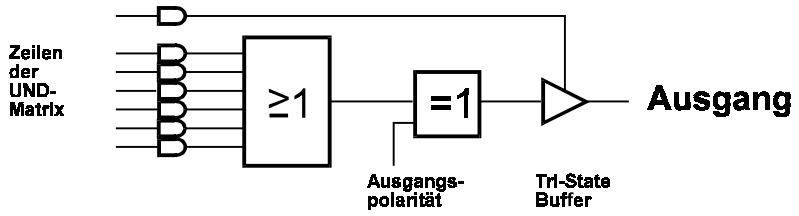


Abb. 2.9 Beispiel-Konfiguration eines Ausgabeblocks

Bei PALs liegt die Konfiguration der Ausgabeblöcke, abhängig vom PAL-Typ, fest und ist nicht veränderbar. Bei einigen Typen sind die Konfigurationen der einzelnen Ausgabeblöcke innerhalb einer PLD verschieden.
Um PLDs noch flexibler zu gestalten, wurde bei den GALs und CPLDs die universelle Ausgangszelle (OLMC - Output Logic Macro Cell, s. Kapitel 4.2.2) eingeführt. Sie ermöglicht es dem Anwender mit Hilfe entsprechender Programmierung eine fast beliebige Konfiguration zu bilden.

2.1.5 Ein praktisches Beispiel

Als praktisches Beispiel soll nun mit der Beispiel-PLD ein RS-Flipflop realisiert werden. Abb. 2.10 zeigt die Lösung mit den intakten „Sicherungen“ (Keise) an den richtigen Kreuzungspunkten.

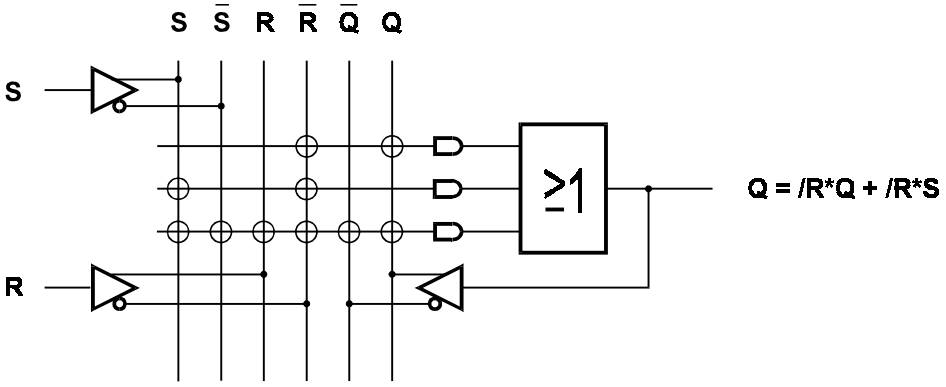


Abb. 2.10 RS-Flipflop realisiert mit der Beispiel-PLD

In diesem Beispiel wird der dritte (hier ist es der unterste) Produktterm nicht benötigt. Durch das Und-verknüpfen der Signale mit deren Inversion bleibt er inaktiv.

2.2 Programmiertechnologien

Damit die Programmierdaten dauerhaft im PLD-IC gespeichert bleiben, müssen sie im IC durch programmierbare Elemente (Zellen) repräsentiert sein. Diese Elemente befinden sich, bildlich betrachtet, in der UND-Matrix und in den Macro-Zellen. Die programmierbaren Elemente stellen im unprogrammierten Zustand eine elektrisch leitende Verbindung zwischen zwei Punkten dar. Wird ein Element programmiert, so ist die entsprechende Verbindung unterbrochen (Ausnahme: Anti-Fuse).In der Praxis finden verschiedene Techniken Verwendung, um derartige Elemente zu realisieren.
Im Folgenden werden diese Techniken näher erläutert:

- **FUSE-Technologie**
  Während der Programmierung werden Silizium-Brücken auf dem Chip durch einen hohen Stromimpuls zerstört. Nach der Programmierung ist die Verbindung getrennt. Die richtige Wahl des Programmierstroms ist bei dieser Technik von wesentlicher Bedeutung. War der Programmierstrom zu gering, kann es u.U. vorkommen, daß die Trennstelle durch thermische Ausdehnung wieder leitend wird. War er zu groß, dann können Materialspritzer den Baustein unbrauchbar machen.Vertreten bei PROM, PAL, FPLA, FPGA, FPLS.

- **ANTI-FUSE-Technologie**
  Eine Anti-Fuse ist einem Kondensator ähnlich (Dielektrikum zwischen zwei Metall-Flächen) aufgebaut und hat eine Größe von einigen Mikrometern. Die Anti-Fuse hat im unprogrammierten Zustand einen sehr hohen Widerstand. Beim Programmieren wird eine relativ hohe Spannung an die „Platten“ gelegt und solange gewartet, bis sich ein definierter Strom einstellt. Der „Kondensator“ schlägt durch und bildet eine leitende Verbindung zwischen den beiden Metallflächen. Nach der Programmierung ist die Verbindung hergestellt. Diese Technologie ist, mit unterschiedlichem Aufbau, vertreten bei QuickLogic unter dem Namen ViaLink und bei ACTEL unter dem Namen PLICE.

- **„Floating-Gate“-ECMOS-Technologie**
  Als Schaltelement (Zelle) arbeitet hier ein CMOS-Transistor mit einem elektrisch isoliert angebrachtem Gate. Im unprogrammierten Zustand stellt er eine Verbindung her. Beim Programmiervorgang werden mit Hilfe des sogenannten Tunneleffekts Ladungsträger in den Gate-Bereich „gepumpt“. Um diesen Effekt zu erreichen, bedarf es einer Programmierspannung von ca. 12 V. Die Ladungsträger bestimmen den Schaltzustand des Transistors und sind durch die sie umgebende Isolation für viele Jahre im Gate gebunden. Da der Tunnelvorgang physikalische Beschädigungen an der Isolation erzeugt, kann eine Programmierung nicht beliebig oft erfolgen. Beim Löschvorgang werden die Ladungsträger mittels UV-Licht aus dem Gate-Bereich entfernt. Bausteine die mit dieser Technologie arbeiten sind am Glasfenster im Baustein zu erkennen (vgl. EPROM).

- **EEPROM-Technologie**
  Hier gilt das Gleiche wie bei der EPROM-Technologie, lediglich beim Löschvorgang besteht ein Unterschied. Er erfolgt wie auch der Programmiervorgang auf elektronischem Wege durch Programmierimpulse. Es werden laut Hersteller mindestens 100 Schreib-/Lösch-Zyklen gewährleistet.

- **SRAM-Technologie**
  Ein Flipflop (SRAM-Zelle) arbeitet hier als Schaltelement. Da nach dem Abschalten der Versorgungsspannung die programmierten Zustände „verloren" gehen, muß dementsprechend Vorsorge getroffen werden. Dies geschieht meist durch ein zusätzliches EPROM, das die Programmierdaten enthält. Beim Anlegen der Versorgungsspannung liest eine PLD-interne Logik die Daten aus dem EPROM und schreibt sie in das PLD-interne SRAM.

- **FLASH-Technologie**
  Flash-Speicherelemente verbinden die besonderen Merkmale von EPROM- und EEPROM-Technologie. Sie sind elektrisch programmierbar und auch elektrisch löschbar, besitzen eine wesentlich höhere Integrationsdichte als EEPROM-Elemente, und es werden mindestens 10.000 Schreib-/Lösch-Zyklen gewährleistet.

2.3 Vor- und Nachteile von PLDs

Vorteile:

+ Flexibilität
  Durch ihren internen Aufbau sind PLDs wesentlich flexibler als Standardlogik. Dadurch ist es möglich, Funktionen zu integrieren, die in Standardbausteinen nicht zu finden sind.

+ Platzersparnis
  Eine Vielzahl an implementierbaren Logikfunktionen ermöglicht, flächenintensive Logik einfach und platzsparend zu realisieren.

+ Kosteneinsparung
  Platz (Flächen) - und Baustein-Einsparungen führen zu geringeren Kosten.

+ Erhöhte Betriebssicherheit
  Ein Wegfall von Bausteinen und somit auch Leiterbahnen erhöht die Betriebssicherheit (Stichwort EMV) und die Zuverlässigkeit/Fehlerwahrscheinlichkeit.

+ Erleichterungen beim CAD
  Die Entflechtung einer Platine wird durch das relativ frei wählbare Pin-Layout bei den PLDs erleichtert. Nachträgliche Funktionsänderungen sind meistens ohne Layout-Änderungen möglich.

+ EECMOS und ECMOS Technologien (wird nicht von allen PLDs verwendet !)
  Sie kombinieren den CMOS-Prozeß mit elektrisch- oder UV-Licht-löschbarer Speichertechnologie und ermöglichen ein mehrmaliges Programmieren. Dies ist ein Vorteil, der besonders in der Entwicklungsphase zum Tragen kommt.

+ Kopierschutz
  Spezielle Schutzeinrichtungen verhindern ein unerlaubtes Auslesen der Programmierdaten.

Nachteile:

– Kosten
  Design-Werkzeuge (Software) und ggf. Programmiergeräte verursachen zusätzliche Kosten. Die Software setzt das Vorhandensein eines Computers voraus.

– Stromaufnahme
  Zur Zeit haben PLDs noch eine relativ hohe Stromaufnahme (einige 10mA), die jedoch anwendungsabhängig durch die höhere Integrationsdichte kompensiert werden kann. Die Stromaufnahme ist u.a. abhängig von der Bausteinauslastung, der gewählten Geschwindigkeitsklasse und den evtl. verwendeten Taktfrequenzen.

2.4 Unterscheidungsmerkmale programmierbarer Logik

Der Begriff PLD ist grundsätzlich für alle vom Anwender programmierbaren Logikbausteine gültig. Nicht nur PALs und GALs gehören zur Gruppe der PLDs, sondern auch noch weitaus mehr Bausteine mit unterschiedlichen Eigenschaften. Einige Unterscheidungsmerkmale von PLDs sind:

- interne Struktur
- verwendete Programmmier-Technologie
- PLD ist nur einmal programmierbar (OTP) oder wiederprogrammierbar
- Geschwindigkeits-Klasse
- Stromaufnahme
- Versorgungsspannung (3,3 oder 5V)
- universelle- oder fest konfigurierte Ausgangszelle
- Komplexität

Abhängig von ihrer internen Struktur (Architektur) werden PLDs in die Gruppen PAL/GAL, CPLD und FPGA unterteilt.
Die Gruppe der PALs und GALs (< 1000 Gatter) wird manchmal auch als SPLD (Simple PLD) oder Low Density PLD bezeichnet.
CPLDs (> 1000 Gatter) sind Bausteine mit PAL-ähnlichen Teilstrukturen. Sie besitzen eine sogenannte Multiple Array Structure, d.h. es befinden sich viele kleine, PAL-ähnliche Blöcke mit teilweise großer Integrationsdichte auf dem Chip, die über eine Schalt- oder Switch-Matrix miteinander verbunden sind.
Bei den FPGAs sind viele, sehr kleine Module mit geringer Logikintegration in einer Art Matrix auf dem Chip plaziert. In den Zwischenräumen liegen die sogenannten Routing Channels, in denen die Verdrahtung der Logikmodule durchgeführt wird.

	PAL/GAL Architektur	**CPLD Architektur**	**FPGA Architektur**
Logik-Ressourcen	Wenige, sehr einfache Ressourcen	Wenige, komplexe Ressourcen	Viele, sehr einfache Elemente Umsetzung von komplexer Logik erfordert viele Module
Verbindungsaufbau	Verdrahtungsmatrix	Schaltmatrix	Logik kann global und lokal genutzt werden
Geschwindigkeit	Sehr schnell Langsam beim Einsatz von Rückkopplungen	Schnell	Abhängig von Verdrahtung und Plazierung
Zeitliches Verhalten	Vorhersagbar	Vorhersagbar	Nicht Vorhersagbar
Software	PLD-Compiler	Mapper/Fitter	Place and Route, Simulation

Abb 2.11 Unterschied zwischen PAL/GAL, CPLD und FPGA

2.4.1 Abkürzungen

Tabelle 1. Zusammenstellung der gebräuchlichsten Abkürzungen

In dieser Tabelle (sie erhebt keinen Anspruch auf Vollständigkeit) sind die wichtigsten
Abkürzungen aus dem Bereich der programmierbaren Logikbausteine zusammengestellt.

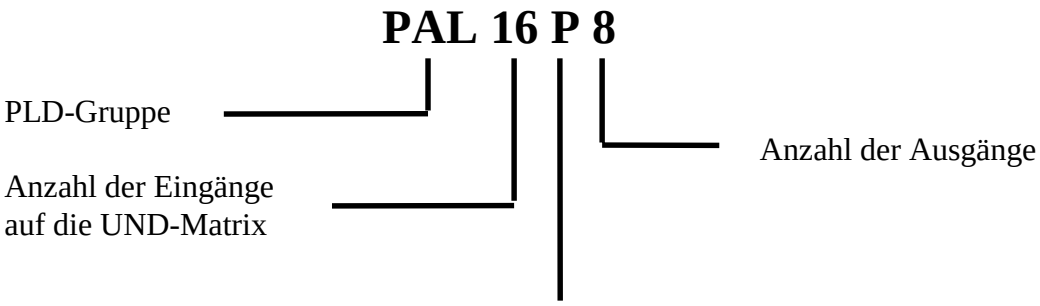
• ASIC	Application Specified lntegrated Circuit	Anwendungsspezifisches IC
• CPLD	Complex PLD	Komplexer PLD
• EEPLD	Electrically Erasable Programmable Logic Device	Elektrisch löschbarer programmierbarer Logikbaustein
• EPLD	(UV-) Erasable Programmable Logic Device	(UV-) Löschbarer programmierbarer Logikbaustein
• FPAD	Field Programmable Address Decoder	Frei programmierbarer Adreßdekoder
• FPAL	Field Programmable Array Logic	Frei programmierbare Logikmatrix
• FPGA	Field Programmable Gate Array	Frei programmierbare Gattermatrix
• FPLA	Field Programmable Logic Array	Frei programmierbare Logikmatrix
• FPLD	Field Programmable Logic Device	Frei programmierbarer Logikbaustein
• FPLS	Field Programmable Logic Sequencer	Frei programmierbare sequentielle Logik
• FPML	Field Programmable Macro Logic	Frei programmierbare Makrologik
• GAL	Generic Array Logic	Universelle Matrix-Logik
• HAL	Hard Array Logic	Vom Hersteller programmiertes PAL
• IFL	lntegrated Fuse Logic	Integrierte Logik mit Trennstellen
• LCA	Logic Cell Array	Logikzellen-Matrix
• PAL	Programmable Array Logic	Programmierbare Logik-Matrix
• PLA	Programmable Logic Array	Programmierbare Logikmatrix
• PLD	Programmable Logic Device	Programmierbarer Logikbaustein
• PLE	Programmable Logic Element	Programmierbares Logikelement
• PLS	Programmable Logic Sequencer	Programmierbare sequentielle Logik
• PML	Programmable Macro Logic	Programmierbare Makrologik

3 PALs

PALs werden in großer Typenvielfalt angeboten. So gibt es PALs mit Registerausgängen, mit EXOR-Gattern anstelle der üblichen Oder-Gatter, mit Komplementär- oder Tristate-Ausgängen. Ferner kann zwischen invertiertem oder nicht invertiertem Ausgang gewählt werden. Um dieses Typenspektrum unterscheidbar zu machen, verwendet man eine kodierte Bezeichnung. Die erste Zahl gibt die Anzahl der Eingänge auf die Logikmatrix an, der mittlere Buchstabe unterscheidet den Ausgangstyp, während die letzte Zahl die Anzahl der Ausgänge angibt (Abb. 3.1).

Trotz der großen Typenvielfalt der PAL-Bausteine gibt es auch einige Nachteile. Als wesentlicher Nachteil erweist sich die Tatsache, daß die PALs nicht elektrisch gelöscht werden können. Dies wirkt sich hauptsächlich auf die Kosten in der Entwicklungsphase aus, in der dann mehrere PALs benötigt werden, falls die Logikfunktionen nicht auf Anhieb richtig definiert sind. Auch der hohe Stromverbrauch von typ. 180 mA ist nicht zu vernachlässigen.

Wenn man für jede Anwendung das optimale PAL einsetzen möchte, benötigt man eine Vielzahl verschiedener Typen. Um die Typenvielfalt zu verkleinern, werden zunehmend GALs mit programmierbaren Ausgangszellen eingesetzt. Die Ausgangszellen sind vom Anwender in verschiedenen Architekturen programmierbar, so daß nur vier GAL-Typen ausreichen, um sämtliche PALs zu ersetzen.



Abk.	Ausgangs-Typ
C	Komplementärausgang
H	Ausgang aktiv HIGH
L	Ausgang aktiv LOW
P	Ausgang mit program-mierbarer Polarität
R	Registerausgang
X	EXOR-Gatter und Registerausgang

Abb 3.1 Typenbezeichnung bei PALs

4 GALs

4.1 Was sind GALs ?

GALs sind eine Weiterentwicklung der PALs. Durch die eingesetzte EECMOS-Technologie können sie elektronisch gelöscht und bis zu hundertmal neu programmiert werden. Die GALs 16V8 und 20V8 besitzen je acht konfigurierbare, universelle Ausgangszellen (OLMC). Der Begriff „konfigurierbar“ bedeutet hier, daß der Anwender verschiedene Ausgangsstrukturen festlegen kann. Die Ausgänge können entweder als kombinatorischer Ausgang, Tristate-Ausgang oder Registerausgang definiert werden. In Abb. 4.1 ist die Typenbezeichnung der GALs, hier beispielhaft anhand des GAL20V8, dargestellt. Bei diesem GAL führen 20 Eingänge auf die UND-Matrix. Das „V“ bezeichnet die vielseitig konfigurierbaren Ausgangszellen. Die letzte Zahl gibt die maximale Verzögerungszeit Eingang->Ausgang in Nano-Sekunden an. Nachfolgende Buchstaben geben Auskunft über Stromaufnahme, Gehäuseform und Temperaturbereich des betreffenden GALs.
Abb. 4.2 zeigt die Anschlußbelegung der GAL-Typen 16V8 und 20V8 im Normalbetrieb. Im Programmierbetrieb haben die Anschlüsse eine andere Funktion. Datenblätter dieser beiden GAL-Typen befinden sich im Anhang B.

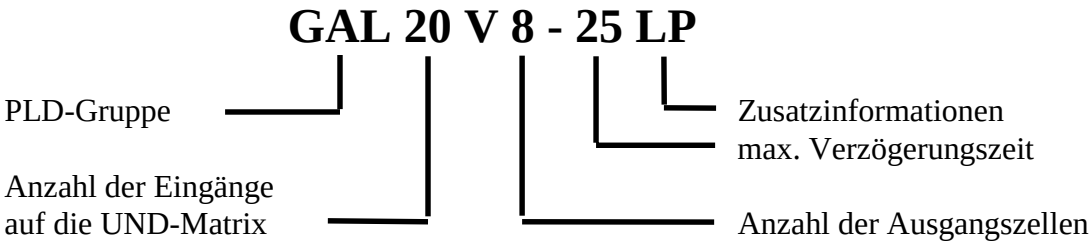


Abb. 4.1 Typenbezeichnung bei GALs

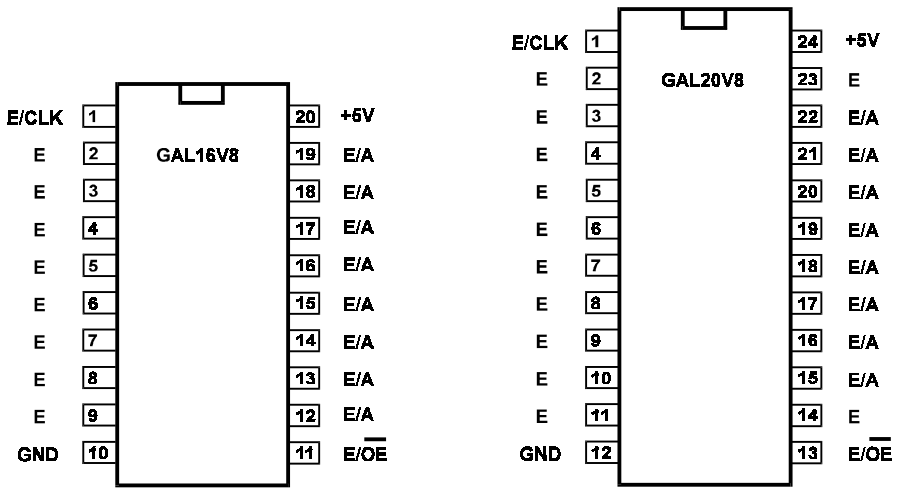


Abb. 4.2 Pinbelegung der GALs 16V8 und 20V8

4.2.1 Logikdiagramm und Funktionsblockschaltbild mit Pinbelegung

Der interne Aufbau von PLDs wird durch das Logikdiagramm (Logic Diagram) und das Funktionsblockschaltbild (Functional Block Diagram) graphisch dargestellt. Für das Funktionsblockschaltbild sind zwei unterschiedliche Darstellungsformen gebräuchlich (vergl. Datenblätter Lattice S. 3-65 und Texas Instruments S. 3).

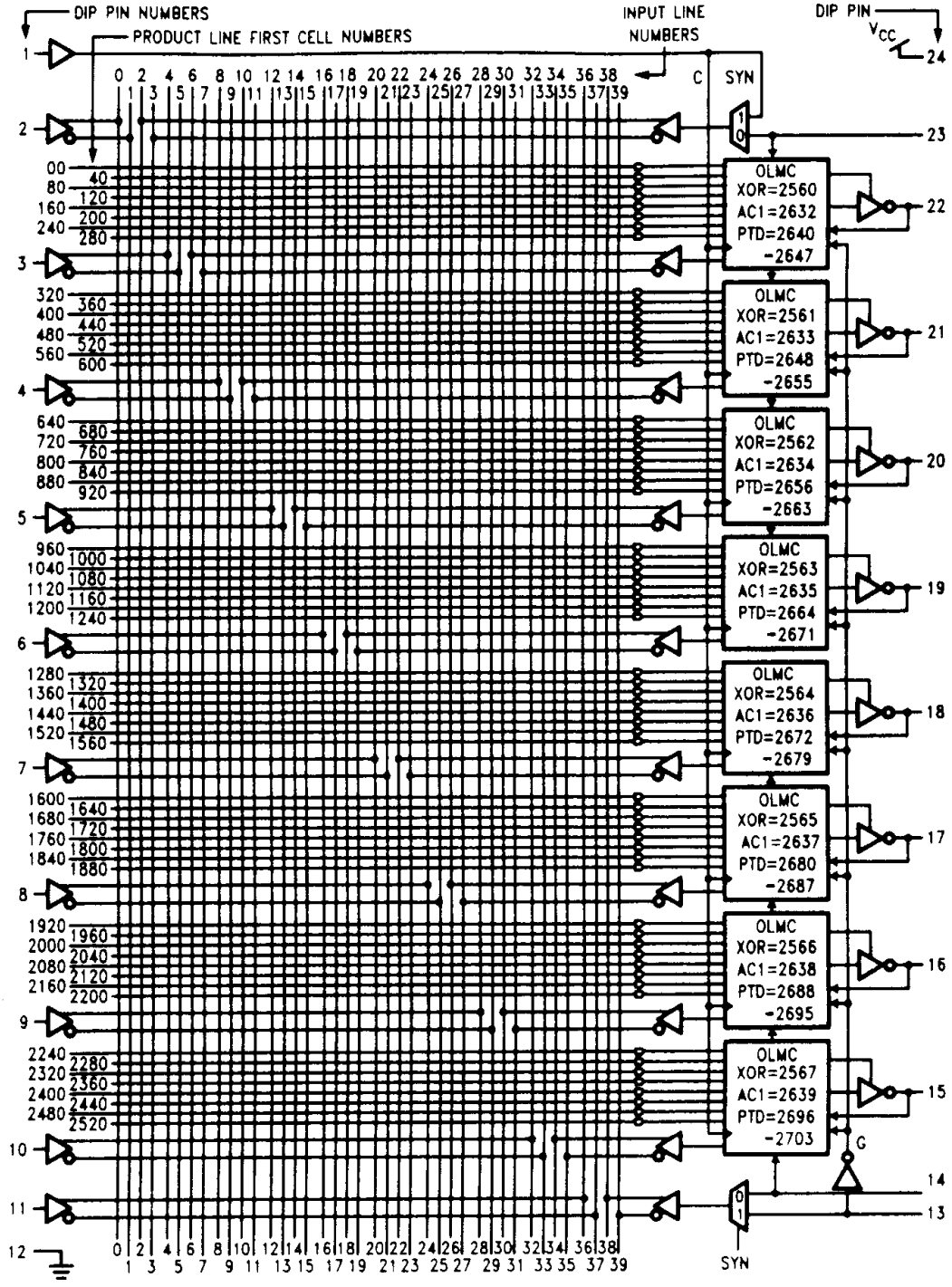


Abb. 4.3 Logikdiagramm des GAL 20V8

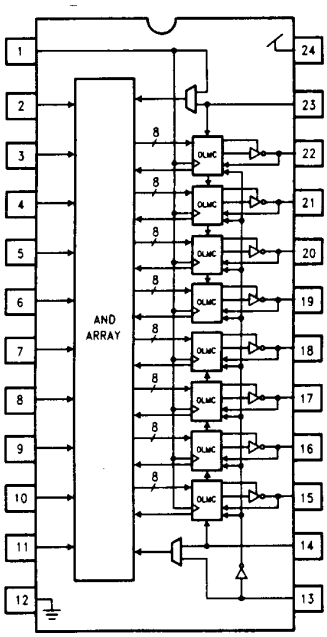


Abb. 4.4 Funktionsblockschaltbild des GAL 20V8

4.2.2 Die universelle Ausgangszelle (OLMC)

Die konfigurierbare Ausgangszelle (OLMC), im Logikdiagramm (Abb 4.3) und im Funktionsblockschaltbild (Abb 4.4) durch einen Kasten symbolisiert, ist in Abb. 4.5 ausführlich dargestellt.

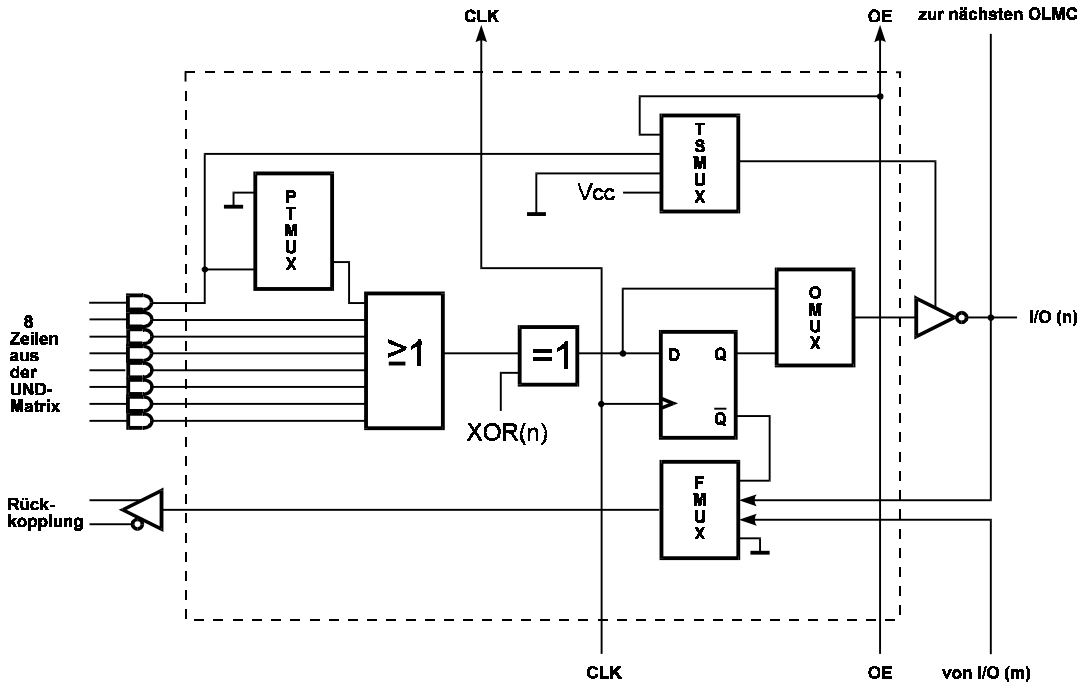


Abb. 4.5 Die konfigurierbare Ausgangszelle (OLMC)

Die verschiedenen Konfigurationen der OLMC werden durch die programmierbaren Steuersignale SYN, AC0, AC1(n) und XOR(n) eingestellt. Diese Steuersignale sind programmierbare Bits im sogenannten Architektur-Steuerwort (Architecture Control Word, ACW), einem definierten Speicherbereich im Baustein. Der Anwender braucht sich während des GAL-Entwurfs über diese Signale keine Gedanken zu machen, da die Software alle Einstellungen automatisch vornimmt. Für das Verständnis der OLMC ist es aber notwendig, näher auf diese Signale einzugehen.
SYN und AC0 definieren die Ausgangszellen global, während AC1(n) und XOR(n) jede OLMC individuell konfigurieren, wobei „n“ die Nummer der OLMC angibt.

- Das XOR(n) Signal wirkt auf das XOR-Gatter der OLMC(n) und bestimmt damit die Ausgangspolarität entweder zu aktiv LOW (XOR(n)=0) oder zu aktiv HIGH (XOR(n)=1).
- AC0 und AC1(n) steuern die vier Multiplexer und legen so die Ausgangsstruktur der einzelnen Zellen fest. Es folgen die Multiplexer im Einzelnen: Produktterm (PTMUX) -, Tristate (TSMUX) -, Ausgangs (OMUX) - und Rückkopplungs (FMUX) - Multiplexer.
- SYN bestimmt, ob das GAL kombinatorisch oder sequentiell arbeitet. Bei sequentieller Arbeitsweise (SYN = 0) ist Pin 1 der gemeinsame Takteingang der internen D-Flipflops. Pin 13 (GAL 20V8) bzw. Pin 11 (GAL 16V8) stellt den gemeinsamen Aktivierungseingang /(Output Enable) der Registerausgänge dar. Bei SYN=1 sind diese Pins normale Eingänge, deren Signale in die Spalten der UND-Matrix führen.

OLMC ist konfiguriert als	SYN	AC0	AC1(n)	GAL ist im
Eingang	1	0	1	Betriebsmodus 1
kombinatorischer Ausgang	1	0	0	
Tristate-Ausgang	1	1	1	Betriebsmodus 2
Tristate-Ausgang	0	1	1	Betriebsmodus 3
Register-Ausgang	0	1	0	

Abb. 4.6 Die fünf möglichen Konfigurationen der OLMC

Die Abbildungen 4.7 bis 4.11 zeigen die fünf möglichen Konfigurationen der OLMC.

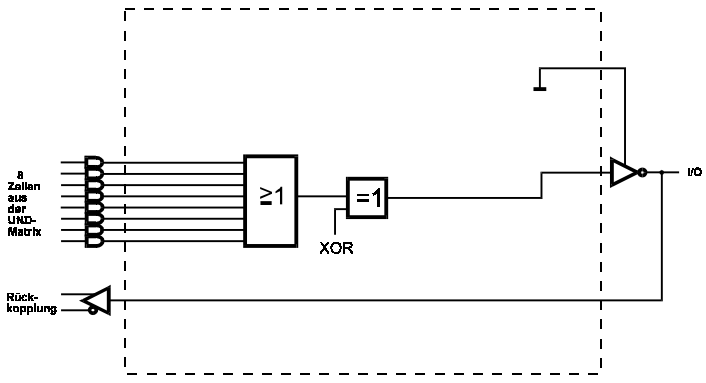


Abb. 4.7 Die OLMC konfiguriert als Eingang (s. Fußnote 1)

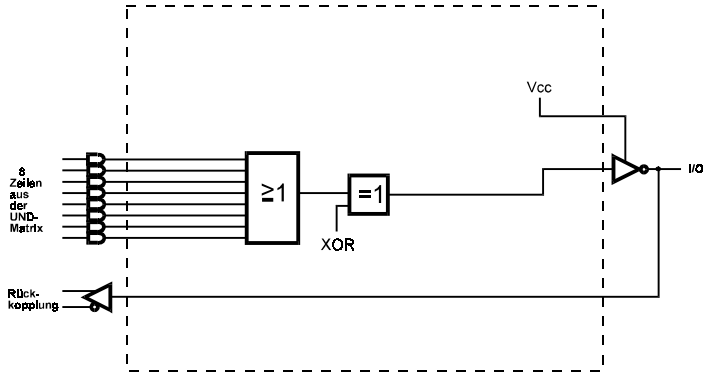


Abb. 4.8 Die OLMC konfiguriert als kombinatorischer Ausgang[1]

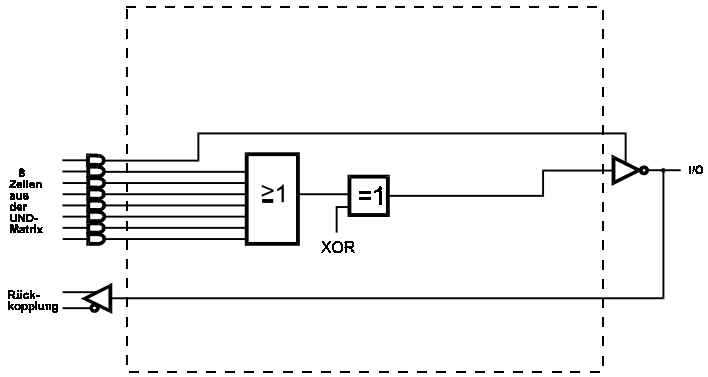


Abb. 4.9 Die OLMC konfiguriert als Tristate-Ausgang[2]

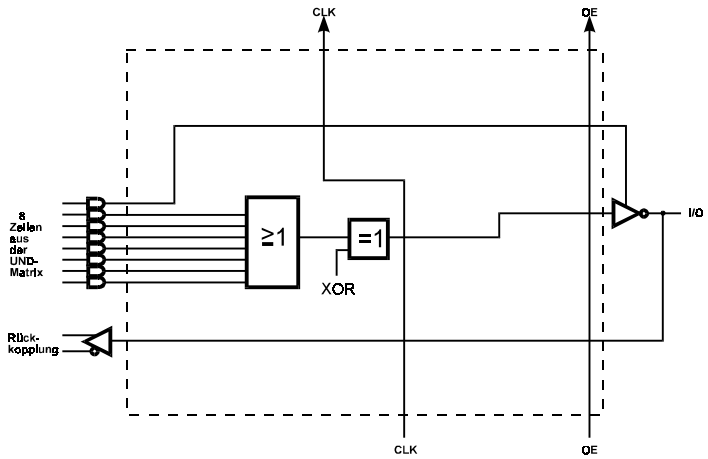


Abb. 4.10 Die OLMC konfiguriert als Tristate-Ausgang im Registermodus

[1] Keine Rückkopplung von Pin 15 und 16 beim GAL16V8 bzw. Pin 18 und 19 beim GAL20V8
[2] Keine Rückkopplung von Pin 12 und 19 beim GAL16V8 bzw. Pin 15 und 22 beim GAL20V8

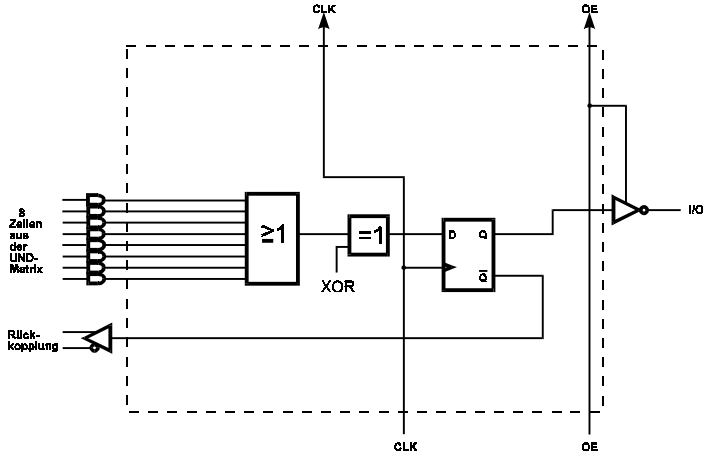
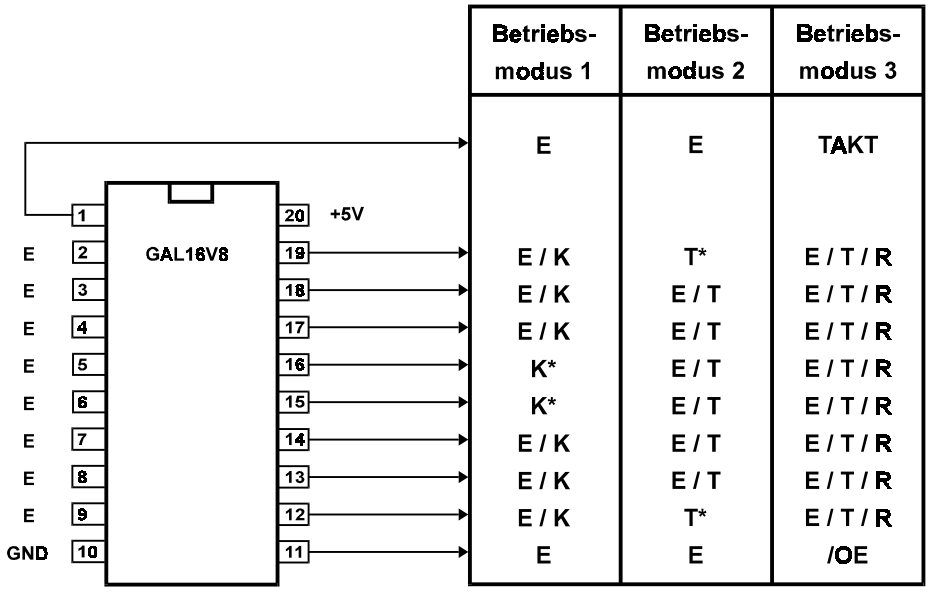


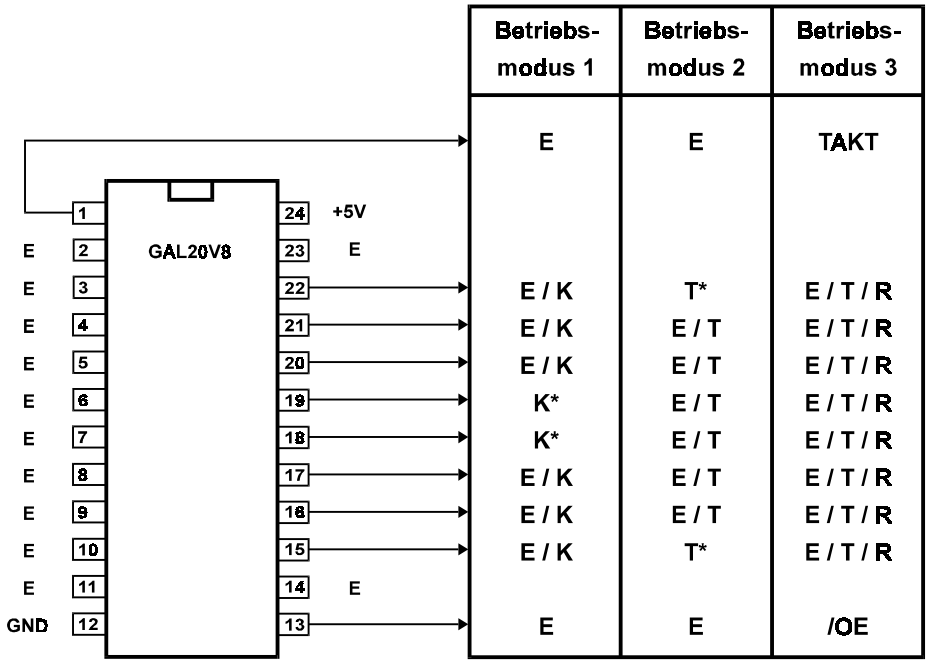
Abb 4.11 Die OLMC konfiguriert als Registerausgang

4.3 Die drei Betriebsmodi

Wie aus Abb. 4.6 ersichtlich ist, sind grundsätzlich drei Betriebsmodi möglich. Abb. 4.12 zeigt die Pinbelegung der beiden Standardtypen GAL 16V8 und GAL 20V8 für alle drei Betriebsmodi. Daraus geht hervor, welche Ausgangsstruktur der Anwender in den einzelnen Betriebsmodi wählen kann.



Betriebs-modus 1	Betriebs-modus 2	Betriebs-modus 3
E	E	TAKT
E / K	T*	E / T / R
E / K	E / T	E / T / R
E / K	E / T	E / T / R
K*	E / T	E / T / R
K*	E / T	E / T / R
E / K	E / T	E / T / R
E / K	E / T	E / T / R
E / K	T*	E / T / R
E	E	/OE



Betriebs-modus 1	Betriebs-modus 2	Betriebs-modus 3
E	E	TAKT
E / K	T*	E / T / R
E / K	E / T	E / T / R
E / K	E / T	E / T / R
K*	E / T	E / T / R
K*	E / T	E / T / R
E / K	E / T	E / T / R
E / K	E / T	E / T / R
E / K	T*	E / T / R
E	E	/OE

Abkürzung	Erläuterung
E	Eingang
K	Kombinatorischer Ausgang mit Rückkopplung
/OE	/ (OUTPUT ENABLE) für Registerausgänge
TAKT	Taktsignal für Registerausgänge
T	Tristate-Ausgang mit Rückkopplung
*****	Ausgang ohne Rückkopplung !
R	Registerausgang

Abb. 4.12 Pinbelegung der GALs in den drei Betriebsmodi

4.3.1 Kombinatorischer Ausgang (Betriebsmodus 1)

In Betriebsmodus 1 können alle acht Ausgangszellen als kombinatorische Ausgänge konfiguriert werden. Ein kombinatorischer Ausgang ist ständig aktiv, d.h. er kann nicht den hochohmigen Zustand einnehmen. Die Ausgangssignale der Pins 18 und 19 beim GAL20V8 bzw. der Pins 15 und 16 beim GAL16V8 werden in diesem Modus **nicht** zurückgekoppelt ! Aufschluß darüber, welche Ausgänge zurückgekoppelt werden, gibt Abb. 4.12. Wird eine OLMC als Eingang definiert, so wird der entsprechende Tristate-Treiber „hochohmig“ geschaltet. Das Eingangssignal einer als Eingang konfigurierten OLMC gelangt über den Rückkopplungsmultiplexer (FMUX) in die UND-Matrix. Für jeden Ausgang kann eine Logikgleichung definiert werden, die sich aus maximal acht Produkttermen zusammensetzt. Im folgenden sind nochmals alle Punkte aufgeführt, die für Betriebsmodus 1 von Bedeutung sind.

Betriebsmodus 1 (Simple Mode):

- Die Ausgänge sind ständig aktiv, es ist kein hochohmiger Zustand möglich
- Pro Ausgang können maximal acht Produktterme ODER-verknüpft werden
- Die Eingänge stehen im GAL auch invertiert zur Verfügung
- Produktterme können aus Eingangs- und rückgekoppelten Ausgangssignalen gebildet werden
- Die Ausgangssignale der beiden mittleren OLMCs werden nicht zurückgekoppelt
- Die Ausgänge können wahlweise aktiv HIGH oder aktiv LOW definiert werden

4.3.2 Tristate Ausgang (Betriebsmodus 2)

In Betriebsmodus 2 gelangt ein Ausgangssignal vom Ausgangsmultiplexer über einen Tristate-Treiber zum Ausgangspin. Der Tristate-Treiber kann durch ein Steuersignal aktiviert oder deaktiviert werden. Ist er deaktiviert (Steuersignal=0), so entspricht dies einer Abkopplung der Treiberstufe vom Ausgangspin. Die Treiberstufe ist in diesem Fall hochohmig. Bei den GAL-Bausteinen wird das Steuersignal durch einen Produktterm gebildet. Ergibt die Verknüpfung des Produktterms eine 1, so ist der Ausgang aktiv, während eine 0 den hochohmigen Zustand bewirkt. Da bei den GALs 16V8 und 20V8 für einen Ausgang maximal acht Produktterme vorgesehen sind, stehen zur Bildung der Ausgangsfunktion nur noch sieben Produktterme zur Verfügung. Die Ausgangssignale der Pins 15 und 22 beim GAL20V8 bzw. der Pins 12 und 19 beim GAL16V8 werden in diesem Modus nicht zurückgekoppelt ! Aufschluß darüber, welche Tristate-Ausgänge zurückgekoppelt werden, gibt Abb. 4.12. Soll ein Tristate-Ausgang als Eingang dienen, so ist der Produktterm zur Freigabe des Ausgangs so zu definieren, daß das Ergebnis 0 ergibt. Im folgenden sind nochmals alle wichtigen Punkte zusammengefaßt.

Betriebsmodus 2 (Complex Mode):

- Die Ausgänge sind Tristate-Ausgänge
- Die Freigabe des Ausgangssignals erfolgt durch einen Produktterm
- Pro Ausgang können maximal sieben Produktterme ODER-verknüpft werden
- Produktterme können aus Eingangs- und rückgekoppelten Ausgangssignalen gebildet werden

- Die Ausgänge können wahlweise aktiv HIGH oder aktiv LOW definiert werden
- Die Eingangs- und rückgekoppelten Ausgangssignale stehen im GAL auch invertiert zur Verfügung
- Die Ausgangssignale der beiden äußeren OLMCs werden nicht zurückgekoppelt
- Ein Tristate-Ausgang mit Rückkopplung kann auch als Eingang verwendet werden

4.3.3 Register Ausgang (Betriebsmodus 3)

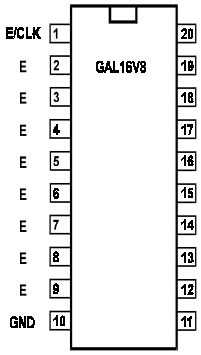
Die GAL-Bausteine verfügen intern über acht D-Flipflops, die wahlweise zwischen das Verknüpfungsergebnis und den eigentlichen Ausgang geschaltet werden können. Ein Ausgang mit vorgeschaltetem D-Flipflop bezeichnet man als Register-Ausgang. Dieses D-Flipflop bewirkt, daß das Ergebnis einer Verknüpfung nicht sofort am Ausgang erscheint, sondern erst mit der positiven Taktflanke des Taktsignales dorthin gelangt. Das Taktsignal wird bei den beiden GALs 16V8 und 20V8 über Pin 1 an die internen D-Flipflops geführt. Man spricht dabei vom synchronen Betrieb, da das Taktsignal allen D-Flipflops gleichzeitig zugeführt wird. Die Register-Ausgänge verfügen darüber hinaus über einen gemeinsamen Aktivierungseingang, der beim GAL16V8 an Pin 11 und beim GAL20V8 an Pin 13 zu erreichen ist. Wird dieses Signal 0, so sind alle Register-Ausgänge aktiviert, während eine 1 den hochohmigen Zustand bewirkt. Die Ausgangssignale aller Register-Ausgänge werden zurückgekoppelt und können somit zur Bildung der Produktterme herangezogen werden. Es lassen sich pro Ausgang acht Produktterme ODER-verknüpfen. Die Register verfügen zusätzlich über einen automatischen Reset beim Einschalten der Betriebsspannung. Ohne Rücksicht auf die gewählte Ausgangspolarität liegen die Register-Ausgänge auf High. Dieser definierte Anfangszustand ist eine große Hilfe beim Design sequentieller Schaltwerke. Man beachte, daß die Ausgangszellen in Betriebsmodus 3 entweder als Register-Ausgang oder Tristate-Ausgang definiert werden können. Zum Beispiel können in diesem Modus alle 8 Ausgänge als Tristate-Ausgänge mit Rückkopplung definiert werden. Auf zwei wichtige Punkte muß noch hingewiesen werden. Das Taktsignal an Pin 1 der GALs sowie der Aktivierungseingang /OE haben keinen Einfluß auf die Tristate-Ausgänge. Die Freigabe des Tristate-Ausgangs erfolgt nach wie vor durch einen Produktterm, nicht durch das Aktivierungssignal /OE. Ferner können diese Signale nicht zur Bildung einer Logikgleichung herangezogen werden.

Betriebsmodus 3 (Registered Mode):

- Ein Ausgang kann wahlweise als Register-Ausgang oder als Tristate-Ausgang konfiguriert werden
- Mit der positiven Taktflanke des Taktsignales an Pin 1 gelangt das Ergebnis der Verknüpfung an den Ausgang (gilt nur für Register-Ausgang)
- Das Ausgangssignal aller Ausgänge wird zurückgekoppelt
- Ein Ausgang kann wahlweise aktiv High oder aktiv Low definiert werden
- Zur Bildung eines Produkttermes können alle Eingänge und alle Ausgangssignale verwendet werden
- Register-Ausgang: 8 Produktterme werden ODER-verküpft
  Tristate-Ausgang: 7 Produktterme werden ODER-verknüpft
- Eine 0 (Low-Pegel) am /OE-Eingang gibt das Ergebnis der Register-Ausgänge frei

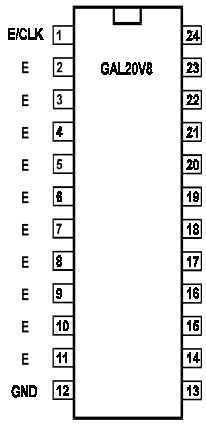
4.4 GALs ersetzen PALs

Da GALs, eine Weiterentwicklung der PALs, mit konfigurierbaren Ausgangszellen ausgestattet sind, reichen zwei GAL-Typen aus, um fast das gesamte Typenspektrum der PALs zu ersetzen. Abb. 4.13 und 4.14 geben einen Überblick darüber, welche PALs ersetzt werden können.



Pin 20	+5V	+5V	+5V	+5V	+5V	+5V	+5V	+5V
Pin 19	A	E	E	E	A	E/A	E/A	A
Pin 18	A	A	E	E	A	A	E/A	E/A
Pin 17	A	A	A	E	A	A	A	E/A
Pin 16	A	A	A	A	A	A	A	E/A
Pin 15	A	A	A	A	A	A	A	E/A
Pin 14	A	A	A	E	A	A	A	E/A
Pin 13	A	A	E	E	A	A	E/A	E/A
Pin 12	A	E	E	E	A	E/A	E/A	A
Pin 11	E	E	E	E	/OE	/OE	/OE	E
PAL	10L8	12L6	14L4	16L2	16R8	16R6	16R4	16L8
PAL	10H8	12H6	14H4	16H2	16RP8	16RP6	16RP4	16H8
PAL	10P8	12P6	14P4	16P2				16P8

Abb. 4.13 GAL16V8 ersetzt 21 PAL-Bausteine



Pin 24	+5V	+5V	+5V	+5V	+5V	+5V	+5V	+5V
Pin 23	E	E	E	E	E	E	E	E
Pin 22	A	E	E	E	A	E/A	E/A	A
Pin 21	A	A	E	E	A	A	E/A	E/A
Pin 20	A	A	A	E	A	A	A	E/A
Pin 19	A	A	A	A	A	A	A	E/A
Pin 18	A	A	A	A	A	A	A	E/A
Pin 17	A	A	A	E	A	A	A	E/A
Pin 16	A	A	E	E	A	A	E/A	E/A
Pin 15	A	E	E	E	A	E/A	E/A	A
Pin 14	E	E	E	E	E	E	E	E
Pin 13	E	E	E	E	/OE	/OE	/OE	E
PAL	14L8	16L6	18L4	20L2	20R8	20R6	20R4	20L8
PAL	14H8	16H6	18H4	20H2	20RP8	20RP6	20RP4	20H8
PAL	14P8	16P6	18P4	20P2				20P8

Abb. 4.14 GAL20V8 ersetzt 21 PAL-Bausteine

Abkürzung	**Erläuterung**
E	Eingang
A	Ausgang
/OE	/ (OUTPUT ENABLE)
E/A	Eingang oder Ausgang
H	Ausgang aktiv HIGH
L	Ausgang aktiv LOW
P	Ausgang mit prog. Polarität
R	Registerausgang

5 Logikentwurf mit ABEL

Beispiel Adreßdekoder:

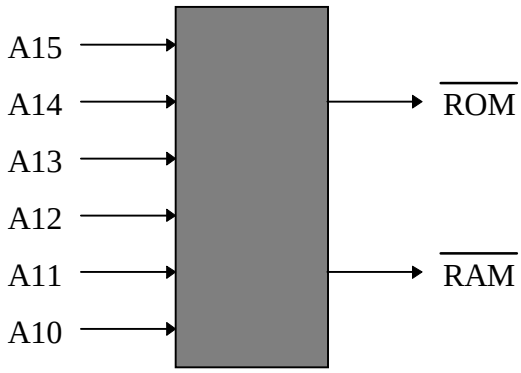


Abbildung 5-1: Das Blockschaltbild

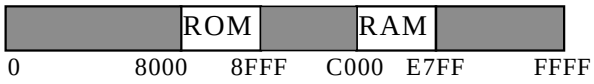


Abbildung 5-2: Aufteilung des Adreßraumes

Die dazugehörige Beschreibung:

```
“Eingange
        A15,A14,A13,A12,A11,A10                 pin 1,2,3,4,5,6;

“Ausgänge
        ROM,RAM                                 pin 15,16 istype ‘com’;

        X = .x. “don’t care

        Adresse = [A15,A14,A13,A12, A11,A10,X,X, X,X,X,X, X,X,X,X];

equations
        !ROM  =  (Adresse >= ^h8000) & (Adresse <= ^h8FFF);
        !RAM  =  (Adresse >= ^hC000) & (Adresse <= ^hE7FF);
end
```

Aus dem Quelltext erzeugte logische Funktionsgleichungen:

```
ROM     = !(  A15 & !A14 & !A13 & !A12 );
RAM     = !(  A15 & A14 & !A13    #   A15 & A14 & !A12 & !A11 );
```

6 Versuchsdurchführung

Schalten Sie den PC ein und starten Sie den PLD-Versuch durch Doppelklicken auf das gleichnamige Icon, nachdem der PC hochgefahren ist. Folgen Sie den Anweisungen des Programms.

Im Fragenteil nach Klicken auf den Ergebnis-Button warten, bis Toolbuch die Antwort ausgewertet hat.

Versuchsteile können wiederholt oder übersprungen werden. Wird ein Fragen- oder Aufgabenteil übersprungen, d.h. bis zum Beenden des Programms nicht bearbeitet, dann hat das natürlich Auswirkungen auf die Bewertung. Sollten Sie den Versuch abbrechen und zu einem anderen Zeitpunkt fortsetzen wollen, dann speichern Sie Ihren Quelltext, sofern Sie sich im letzten Versuchsteil „Praktische Anwendungen“ befinden, auf einer Diskette ab oder sprechen Sie Ihren Betreuer an. Starten Sie zum Fortsetzen den Versuch erneut, geben Sie den gleichen Gruppennamen wie bei der letzten Sitzung an und bearbeiten Sie den Versuch von der Stelle ab, an der Sie abgebrochen haben. Doppelt bearbeitete Versuchsteile werden nicht bewertet. Geben Sie, wenn Sie möchten, am Ende des Versuchs Ihre Kritik und Ihre Anregungen zu diesem Versuch ab. Wählen Sie dazu den entsprechenden Punkt im Menü. Verlassen Sie alle Programme ordnungsgemäß !

Versuchsteil	Bearbeitungsdauer (Richtwert)
Grundlagen und Fragen	1 Stunde
Programmieraufgaben	1,5 Stunden
Praktische Anwendung	1 Stunde

Wichtiger Hinweis zum Versuchsteil „ABEL-Programmieraufgaben“:

Die einführenden Beispiele sind relativ einfach; es ist aber nicht vermeidbar, daß es unterschiedliche richtige Lösungen gibt, die sich z.B. nur in den verwendeten Pins (Pinnummern) unterscheiden. **Halten Sie sich daher unbedingt an Vorgaben im Aufgabentext** (z.B. Pin 5), da der Bewertungsalgorithmus sonst eventuell logisch richtige Lösungen als „falsch“ bewertet. Wenden Sie sich bei mehr als 5 „Falsch“ - Meldungen (pro Programmieraufgabe) an den Betreuer.

7 Literatur und Datenblätter

[1] A. Auer: „PLD-Handbuch“, Hüthig

[2] A. Auer: „Programmierbare Logik-IC“, Hüthig

[3] P. Heusinger: „PLDs und FPGAs in der Praxis“, Franzis-Verlag

[4] D. Bitterle: „GALs:programmierbare Logikbausteine in Theorie und Praxis“, Franzis-Verlag

[5] D.Bitterle: „Schaltungstechnik mit GALs“, Franzis-Verlag

[6] Hack/Hoffmann: „Das GAL Buch“, ELEKTOR-Verlag

[7] Data Book, LATTICE Semiconductor Corporation 1996

[8] Hand Book, LATTICE Semiconductor Corporation 1994

Auf den folgenden Seiten befinden sich Auszüge aus dem LATTICE Data Book für das GAL20V8.