# KAPITEL 2 *Mikroprozessormodell*

## 2.1 Registerwerk

Die CPU benötigt zur Zwischenspeicherung verschiedener Größen einige Speicherplätze:

, Zur Auswahl der Befehle im Programmspeicher oder der Operanden im Datenspeicher benutzt die CPU Adressen, eine Bitkombination bestimmter Breite, die in entsprechenden Speicherplätzen innerhalb der CPU zur Verfügung stehen müssen.

Speicher für Adressen

, Das Programm befindet sich in kodierter Form im Programmspeicher. Die einzelnen Befehle des Programms müssen vor der Ausführung gelesen und interpretiert werden. Hierfür muß dem Operationscode ebenso ein Speicherplatz in der CPU zur Verfügung stehen.

Speicher für Operationscode

, Operanden befinden sich i. Allg. im Datenspeicher des Computers. Zur Bearbeitung müssen sie in die CPU geladen und zwischengespeichert werden. Die Ergebnisse werden dann wieder in den Datenspeicher zurückgeschrieben. Handelt es sich dabei um Zwischenergebnisse, dann ist der Schreibvorgang in den Speicher und der anschließende Ladevorgang in die CPU überflüssig. Zur Speicherung von Operanden und Zwischenergebnissen stellt daher die CPU einige Speicherplätze zur Verfügung.

Speicher für Operanden und Zwischenergebnisse

Diese internen Speicherplätze bezeichnet man als Register und die Gesamtheit der Register als Registerwerk (Register Array: RA). Ein Register sollte folgende Forderungen erfüllen:

Register
Registerwerk RA

- gesteuerte Datenübernahme
- Speicherfähigkeit des Inhalts
- Löschbarkeit des Inhalts
- Datenausgang elektrisch abkoppelbar
- Inkrementieren/Dekrementieren des Speicherinhalts (+1/-1)
- Verschieben (Shiften) des Registerinhalts nach links/rechts

Die beiden letzten Forderungen sind nicht notwendig aber in vielen Fällen wünschenswert.

## 2.1.1 Speicherfähigkeit und gesteuerte Datenübernahme

Als Lösungsansatz für ein Register wird ein synchrones Schaltwerk (s. [23],[24]) benutzt, das mit Hilfe von D-Kippgliedern realisiert wird. Im ersten Schritt beschränke wir uns auf die unbedingt notwendigen Funktionen:

- gesteuerte Datenübernahme
- Speicherfähigkeit des Inhalts

Die Löschbarkeit wird durch die Verwendung eines D-Kippglieds automatisch mitgeliefert. Der Schaltungsentwurf wird für die i-te Stelle des Registers durchgeführt:

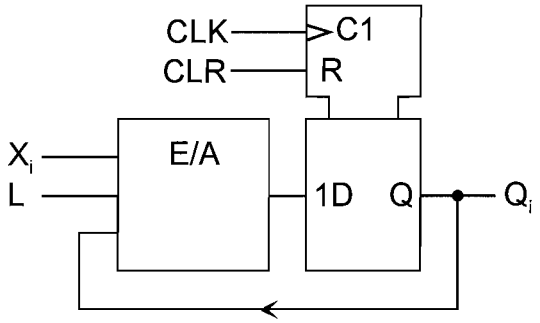



ABB. 2-1: Synchrones Schaltwerk für die i-te Stelle eines Registers

| | |
|---|---|
| $X_i$: | Dateneingang für i-te Stelle |
| $Q_i$: | Datenausgang für i-te Stelle |
| CLK: | Takt zur Synchronisierung |
| CLR: | Löschsignal für das D-Kippglied |
| L: | Steuersignal zum Laden von $X_i$<br>L=0: $Q_i$ bleibt erhalten.<br>L=1: $X_i$ wird bei der nächsten positiven CLK-Flanke in das Register übernommen. |
| E/A: | Eingangsbeschaltung des D-Kippglieds |

Die Funktion des synchronen Schaltwerks wird durch folgende Wahrheitstabelle beschrieben:

| j | | | j+1 |
|---|---|---|---|
| **$X_i$** | **L** | **$Q_i$** | **$D/Q_i$** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

| | $X_i$ | | | |
|---|---|---|---|---|
| L | 1 | 1 | 0 | 0 |
| | 0 | 1 | 1 | 0 |
| | | $Q_i^j$ | | |

Karnaugh-Diagramm

TABELLE 2-1: Wahrheitstabelle mit Karnaugh-Diagramm eines 1-Bit-Registers

Die algebraische Gleichung für das Ergebnis $Q_i^{j+1}$ ergibt sich aus dem Karnaugh-Diagramm:

$$Q_i^{j+1} = (Q_i^j \wedge \bar{L}) \vee (X_i \wedge L)$$

Für die Eingangsbeschaltung des D-Kippglieds erhalten wir somit eine sehr einfache Schaltung, die sich als Wahlschalter interpretieren läßt:

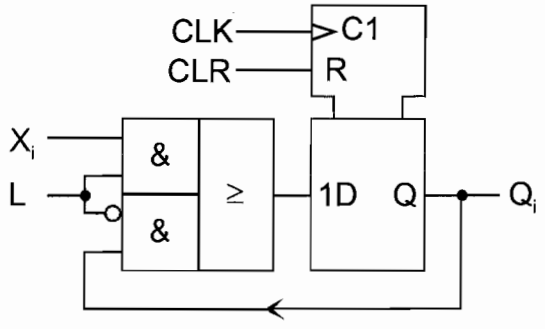



ABB. 2-2: Synchrones Schaltwerk für die i-te Stelle eines Registers mit Eingangsschaltung

Wenn L = 1 ist, dann wird das Eingangssignal durchgeschaltet und mit der nächsten steigenden CLK-Flanke in das D-Kippglied übernommen. Wenn L = 0 ist, dann wird der alte Zustand $Q_i^j$ wieder als neuer Zustand $Q_i^{j+1}$ übernommen, d. h. der Zustand bleibt erhalten.

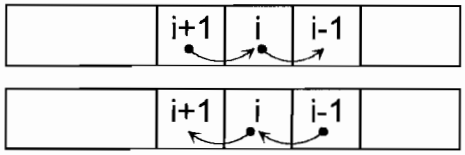
Die Eingangsbeschaltung des D-Kippglieds stellt einen mit L steuerbaren Schalter dar, mit dem zwischen 2 Größen ausgewählt werden kann. Dieses Prinzip läßt sich auf weitere Auswahlmöglichkeiten ausdehnen.

## 2.1.2 Shiftfunktion

Bei der erweiterten Funktion „Shiften rechts " bzw. „Shiften links" gehen die Zustände der Bitzellen in ihre rechten bzw. linken Bitzellen über. Das Steuern der Funktionen nehmen wir mit folgenden Steuersignalen vor:

SHR: Shiften rechts

SHL: Shiften links




Der steuerbare Schalter unseres synchronen Schaltwerks wird jetzt um die Steuersignale SHR und SHL erweitert:

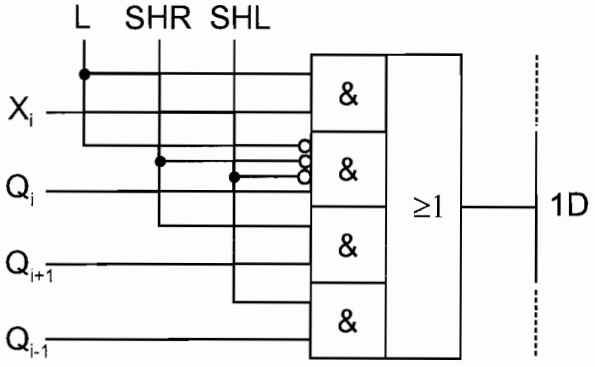



ABB. 2-3: Eingangsbeschaltung mit der Funktionserweiterung „Shiften"

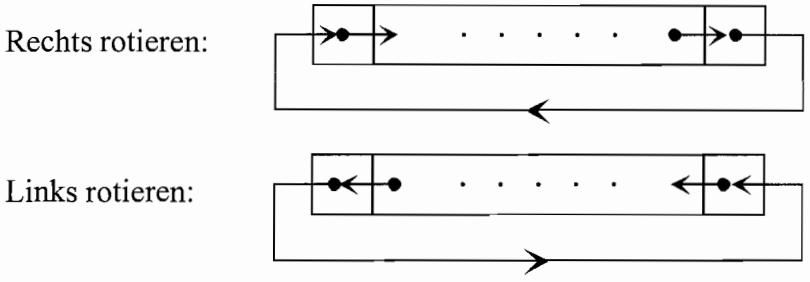
Damit stehen uns folgende Funktionen zur Auswahl:

| L | SHR | SHL | Funktion |
|---|---|---|---|
| 0 | 0 | 0 | Speichern:$Q_i$ bleibt erhalten. |
| 1 | 0 | 0 | Laden:Das Eingangssignal $X_i$ wird geladen. |
| 0 | 1 | 0 | Shiften rechts:Der Zustand $Q_{i+1}$ der linken Nachbarzelle wird übernommen. |
| 0 | 0 | 1 | Shiften links:Der Zustand $Q_{i-1}$ der rechten Nachbarzelle wird übernommen. |

TABELLE 2-2: Registerfunktionen

Bei dieser Lösung wird davon ausgegangen, daß keine widersprüchlichen Steuersignale, z. B. gleichzeitig L = 1 und SHR = 1, anstehen. Bei den Shiftfunktionen muß noch für die Randzellen eine Sonderregelung getroffen werden, da eine direkte Nachbarzelle teilweise nicht vorhanden ist. Mehrere Möglichkeiten sind vorstellbar und werden auch in der Praxis realisiert:

- In die Randzelle wird eine „0" übernommen. Eine Maßnahme, die man häufig für die rechte Randzelle vorsieht. Ein Shiften nach links entspricht dann einer Multiplikation mit 2. Ein Shiften nach rechts entspricht bei vorzeichenfreien Zahlen einer Division durch 2.
- Der Zustand der Randzelle bleibt erhalten. Eine Maßnahme, die man des öfteren bei der linken Randzelle anwendet. Dies entspricht bei vorzeichenbehafteten Zahlen einer Division durch 2.
- Der Zustand der gegenüberliegenden Randzelle wird übernommen. Man bezeichnet diesen Shiftvorgang auch als **Rotieren**: Rotieren




## 2.1.3 Inkrementier- und Dekrementierfunktion

Um schließlich noch die Funktion des Inkrementierens bzw. Dekrementierens zu realisieren, untersuche man am Beispiel dreistelliger Dualzahlen diesen Vorgang. Die Zahl soll bei anstehendem INR-/DCR-Signal bei der nächsten positiven Flanke um 1 erhöht bzw. erniedrigt werden. Beim Inkrementieren wird von 000 beginnend aufwärts gezählt, beim Dekrementieren von 111 beginnend abwärts.

Betrachtet man die Zahlenfolgen, dann läßt sich erkennen, daß die niederwertigste Stelle ihren Zustand bei jeder positiven Flanke des Taktsignals ändert, während die höherwertigen Stellen ihren Zustand nur bedingt ändern. Die zweite Stelle z. B. ändert ihren Zustand nur nach jeder zweiten Flanke, d. h. nur nach jeder zweiten Zustandsänderung der ersten Stelle folgt auch eine in der zweiten Stelle. Diese Zustandsänderung tritt genau dann ein, wenn die niederwertigere Stelle den Wert 1 (INR = 1) bzw. 0 (DCR = 1) hat. Zur Verdeutlichung dieser Situationen sind diese Stellen in der folgenden Abbildung dunkel schattiert hervorgehoben:

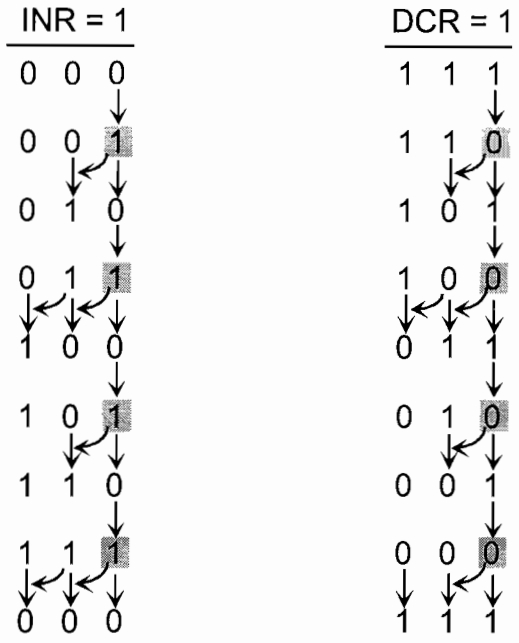



ABB. 2-4: Zahlenfolgen beim Inkrementieren bzw. Dekrementieren

In der dritten Stelle findet man ein ähnliches Verhalten. Eine Zustandsänderung tritt hier nach zweimaliger Zustandänderung der zweiten Stelle ein. Bezieht man außerdem noch die erste Stelle in die Betrachtungen ein, dann zeigt sich, daß dieser Zustandswechsel genau dann erfolgt, wenn beide niederwertigen Stellen den Wert 1 (INR = 1) bzw. 0 (DCR = 1) haben.

Verallgemeinert kann man sagen, die niederwertigste Stelle (i = 0) ändert ihren Zustand $Q_0$ bei jeder positiven Taktflanke, die i-te Stelle (i > 0) wechselt ihren Zustand nur dann, wenn alle niederwertigeren Stellen den Wert 1 (bei INR = 1) bzw. 0 (bei DCR = 1) haben.

Ein Zustandswechsel (Invertieren) kann mit Hilfe eines EXOR-Glieds durchgeführt werden. Das rückgekoppelte Zustandssignal $Q_i$ wird nicht direkt, sondern über ein EXOR-Glied auf die Eingangsbeschaltung des synchronen Schaltwerks geführt, die Invertierung wird mit Hilfe eines Steuersignals $CPL_i$ (complement) gesteuert:

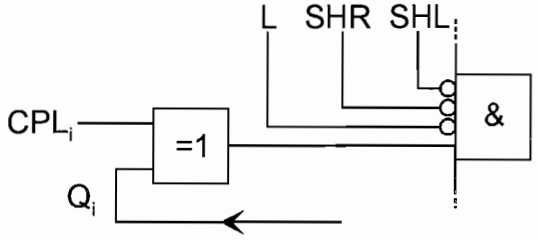



ABB. 2-5: Invertierung des rückgekoppelten Zustandsignals

Das Invertierungssignal $CPL_0$ für das Signal $Q_0$ wird direkt durch eine ODER-Verknüpfung von INR und DCR gebildet:

$$CPL_0 = (INR \vee DCR)$$

Das Inkrementsignal $INR_i$ bzw. das Dekrementsignal $DCR_i$ für die Zustandssignale $Q_i$ der höherwertigen Stellen ($i > 0$) unterliegt noch der Nebenbedingung, daß die Zustandssignale $Q_{i-1}$ bis $Q_0$ 1 (INR = 1) bzw. 0 (DCR = 1) sind:

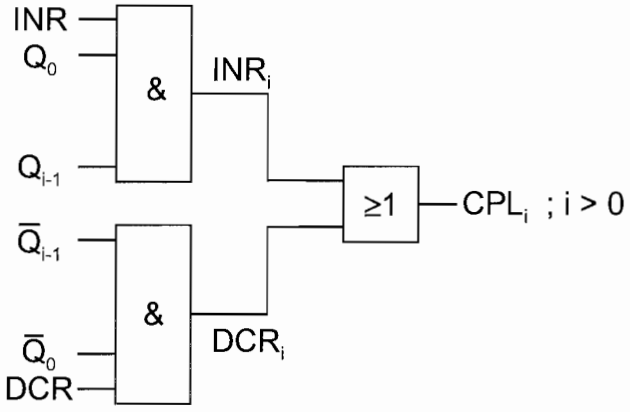



ABB. 2-6: Bedingter Zustandswechsel

Diese Schaltung wiederholt sich im Prinzip an jeder Stelle des Registers. Lediglich die Anzahl der zu verknüpfenden Signale nimmt von Stelle zu Stelle zu. Es genügt daher, die in der vorherigen Stelle gebildeten Signale $INR_{i-1}$ und $DCR_{i-1}$ wieder zu verwerten und lediglich mit $Q_{i-1}$ bzw. $\neg Q_{i-1}$ einfach UND zu verknüpfen. Dies führt dann zu der vereinfachten Schaltung für die Inkrement-/Dekrement-Funktion für die i-te Stelle:

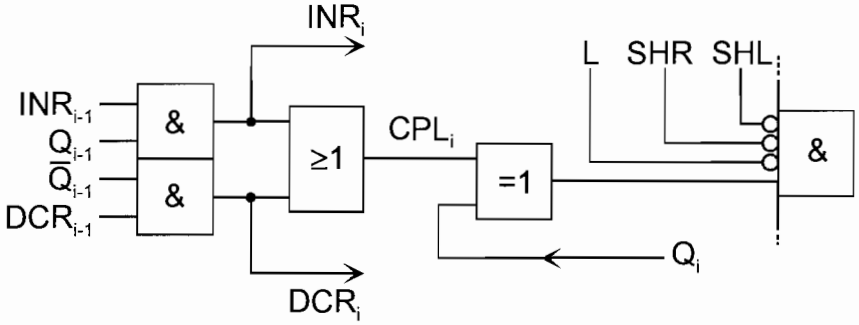



ABB. 2-7: Steuerlogik für die Inkrement-/Dekrement-Funktion der i-ten Stelle eines Registers

Diese Schaltung ist für alle höherwertigen Stellen (i > 0) entwickelt worden, sie berücksichtigt den bedingten Zustandswechsel. Will man die Schaltung in Abbildung 2-7 auch für die 0-te Stufe mit ihrem unbedingten Zustandswechsel anwenden, so muß noch folgende Festlegung getroffen werden:

$$INR_{-1} = INR$$
$$DCR_{-1} = DCR$$
$$Q_{-1} = 1$$
$$\neg Q_{-1} = 1$$

Hierbei wird einerseits berücksichtigt, daß die 0-te Stelle keine Vorgängerstelle hat (i = -1) und der Zustandswechsel unbedingt zu erfolgen hat.

## 2.1.4 Tristate-Ausgang

Bei der Realisierung des Registerwerks ist bisher noch eine Forderung unberücksichtigt geblieben: die Forderung nach der elektrischen Abschaltbarkeit des Ausgangs. Diese Möglichkeit, den Ausgang abzuschalten, benötigt man, wenn mehrere Register zur Datenübertragung miteinander verbunden werden müssen.

Ein Datenaustausch zwischen zwei Partnern ist noch ohne Zusatzaufwand möglich, der Ausgang des einen wird einfach direkt mit dem Eingang des anderen verbunden:

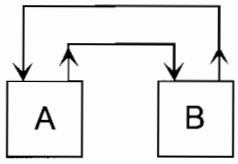



ABB. 2-8: Direkte Kopplung der Register A und B

Erweitert man die Anzahl der Register auf drei, dann führt die Notwendigkeit, jeden mit jedem kommunizieren zu lassen, zu Problemen:

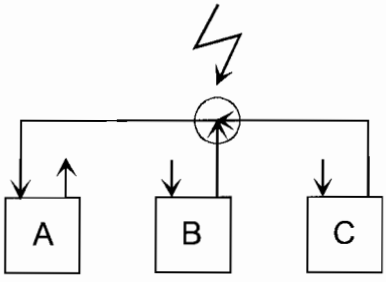



ABB. 2-9: Konfliktsituation durch direkte Verbindung zweier Ausgänge

So muß z. B. der Eingang von A mit den beiden Ausgängen von B und C verbunden werden. Damit entsteht ein Konflikt zwischen den Ausgängen von B und C. Eine sehr einfache Lösung dieses Problems erreicht man dadurch, daß man alle Ausgänge über einen steuerbaren Schalter abkoppelt. Die Ausgänge der Schalter und die Eingangsleitungen können nun alle über eine gemeinsame Leitung miteinander verbunden werden. Bei einer Datenübertragung wird nur das Senderegister über den geschlossenen Ausgangsschalter auf die gemeinsame Datenleitung gekoppelt. Eine Datenübertragung kann nun von dem Senderegister zu einem oder mehreren Empfangsregistern stattfinden, ohne daß eine Konfliktsituation entsteht:

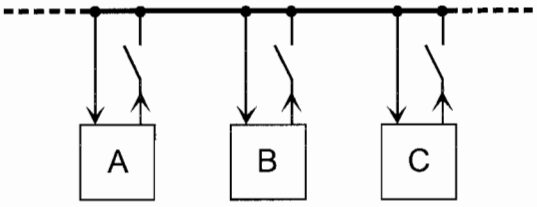



ABB. 2-10: Register mit steuerbarem Ausgang und gemeinsamer Datenleitung

Voraussetzung für diese konfliktfreie Übertragung ist die Möglichkeit, die Schalter entsprechend steuern zu können. Diese Schalterfunktion läßt sich mit folgender Transistor-Ausgangsstufe realisieren:

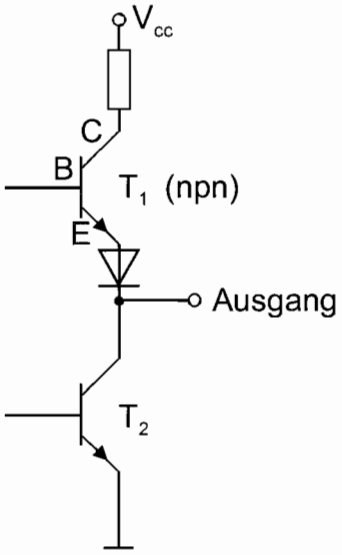



ABB. 2-11: Tristate-Ausgangsstufe

Die Diode dient als Hubdiode. Der Ausgang der Transistorschaltung kann je nach Schaltzustand der Transistoren 3 Zustände annehmen:

| $T_1$ | $T_2$ | Ausgang |
|---|---|---|
| leitend | gesperrt | high |
| gesperrt | leitend | low |
| gesperrt | gesperrt | potentialfrei: Tristate |

TABELLE 2-3: Tristate-Zustände

Für die Steuerung des Tristate-Ausgangs benutzt man ein Steuersignal „ENABLE", abgekürzt EN. Der Tristate-Ausgang wird durch folgende Symbole dargestellt:

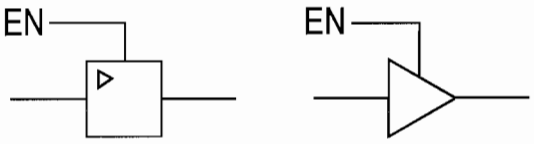
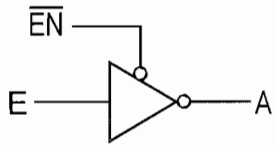



ABB. 2-12: Tristate-Treiber

Als Beispiel für eine invertierende Endstufe betrachte man folgende Transistorschaltung mit Tristate-Ausgang:



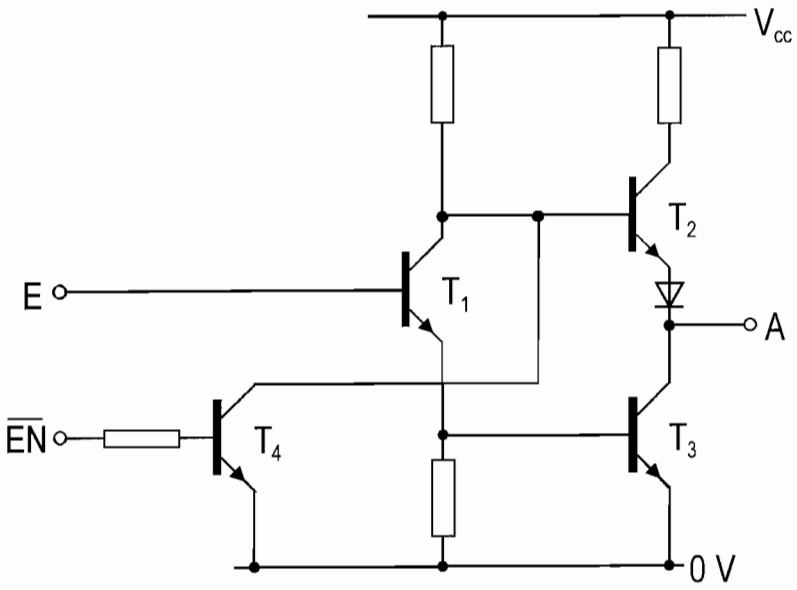




ABB. 2-13: Schaltung einer invertierenden Endstufe mit Tristate-Ausgang (Totempfahl-Schaltung)

- Die Hubdiode sorgt dafür, daß T2 sperrt, wenn T1 durchgeschaltet ist.
- Ist ¬EN = 0, dann verhält sich die Schaltung wie eine gewöhnliche Totempfahl-Schaltung.
- Ist ¬EN = 1, so werden die Transistoren T1 bis T3 durch den durchgeschalteten Transistor T4 gesperrt.

## 2.1.5 Register mit Datenbus

Nachdem alle geforderten Funktionen realisiert werden konnten, sollen die entsprechenden Schaltungen zu einem Register integriert werden. Die Schaltungsentwicklungen wurden jeweils für eine Bitstelle des Registers durchgeführt. Zur Integration zu einem n-stelligen Register müssen die einzelnen Bitzellen parallel zusammengefaßt werden. Da die Steuerleitungen der einzelnen Bitzellen identisch sind, werden sie für das Register nur einmal benötigt. Somit ergibt sich ein Schaltwerk mit den in 2.1.1 bis 2.1.4 eingeführten Steuersignalen, n Dateneingängen (Data Input: $DI_0$ bis $DI_{n-1}$) und n Datenausgängen (Data Output: $DO_0$ bis $DO_{n-1}$) :

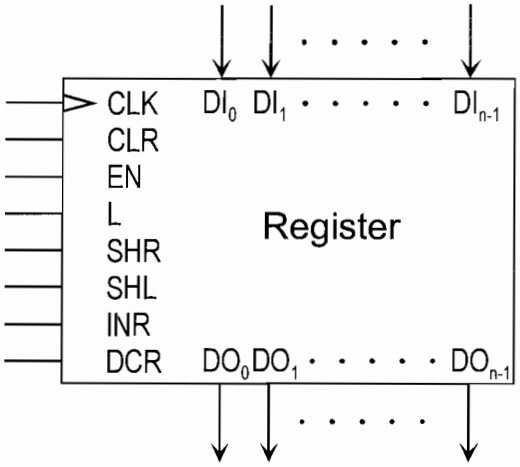



ABB. 2-14: n-stelliges Register

Leitungssysteme, wie z. B. Datenleitungen, die von mehreren Teilnehmern gemeinsam benutzt werden, bezeichnet man als Bus (lat.: omnibus, „für alle"). Werden Signale nur in einer Richtung übertragen, dann handelt es sich um einen unidirektionalen Bus, können die Signale in beide Richtungen übertragen werden, dann spricht man von einem bidirektionalen Bus. Ein Beispiel eines bidirektionalen Datenbusses, bestehend aus einer Busleitung, haben wir bereits in Abbildung 2-10, „Register mit steuerbarem Ausgang und gemeinsamer Datenleitung," auf Seite 2-9 ken-

nengelernt. Solche Bussysteme werden entweder durch Doppellinien oder durch besondere Dicke gegenüber den Einzelleitungen hervorgehoben. Die Übertragungsrichtung kann mit Pfeilen dargestellt werden:

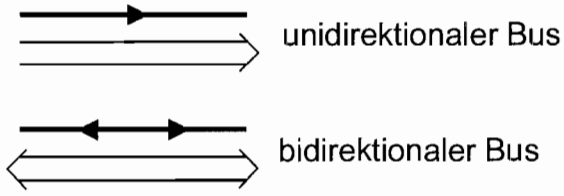



ABB. 2-15: Bussysteme

Entsprechend der Art der Signale spricht man insbesondere von einem

- Datenbus (DB),
- Adreßbus (AB),
- Steuerbus oder Controlbus (CB).

In der obigen Darstellung des Registers können wir somit die einzelnen Datenleitungen durch unidirektionale oder bidirektionale Busse ersetzen:

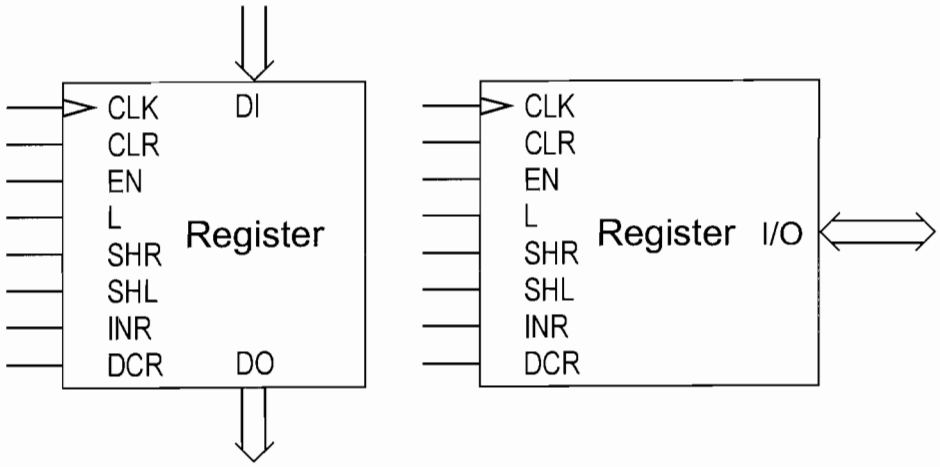



ABB. 2-16: Register mit unidirektionalem Busanschluß bzw. mit bidirektionalem Busanschluß

## 2.1.6 Beispiel eines Registerwerks

Zur Erklärung der Funktionsweise eines Registerwerks betrachten wir ein Registerwerk-Modell, das aus 3 Registern besteht. Die Register A, B und C sind über einen gemeinsamen Datenbus miteinander verbunden, so daß eine Kommunikation zwischen ihnen möglich ist. Aus Gründen der Einfachheit sollen sie nur die notwendigsten Steuersignale besitzen: CLK, L, EN. Da alle Register mit dem selben Takt synchronisiert werden, ist das CLK-Signal nicht gesondert aufgeführt. Mit den Lade- und Freigabesignalen ($L_a$, $E_a$, ...) wird die Datenkommunikation gesteuert.

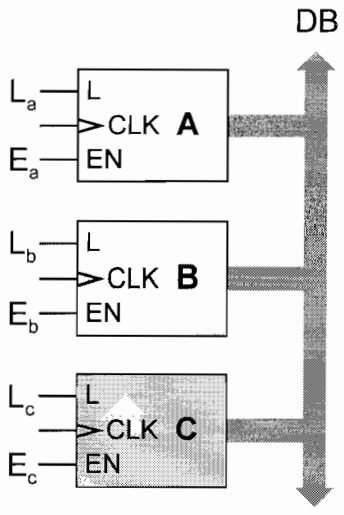



ABB. 2-17: Registerwerk-Modell

Um die Funktionsweise der Datenübertragung zwischen den Registern zu veranschaulichen, sollen die Inhalte der Register A und B miteinander getauscht werden. Da eine gleichzeitige Datenübertragung von zwei Registerinhalten über einen gemeinsamen Datenbus nicht möglich ist, muß der Datenaustausch unter Zuhilfenahme von Register C in mehreren Stufen erfolgen.

Für die Darstellung des Begriffs „Inhalt des Registers R" findet man häufig die Notation:

$$(R) \text{ oder } \langle R \rangle \text{ oder } [R] \text{ oder } \{R\},$$

und dementsprechend für den Ausdruck „Inhalt des Registers $R_1$ in das Register $R_2$ übertragen (kopieren)":

$$(R_1) \rightarrow R_2 \text{ und so fort.}$$

In dieser Abhandlung wird meistens die Darstellungsweise bevorzugt, wie sie bei problemorientierten Sprachen, z. B. bei PASCAL, üblich sind:

$$R_2 := R_1$$

Für die obige Aufgabe muß das Register C als Zwischenregister benutzt werden, um das Überschreiben eines Registerinhalts zu vermeiden. Dementsprechend wird dann die obige Aufgabe in 3 Schritten gelöst:

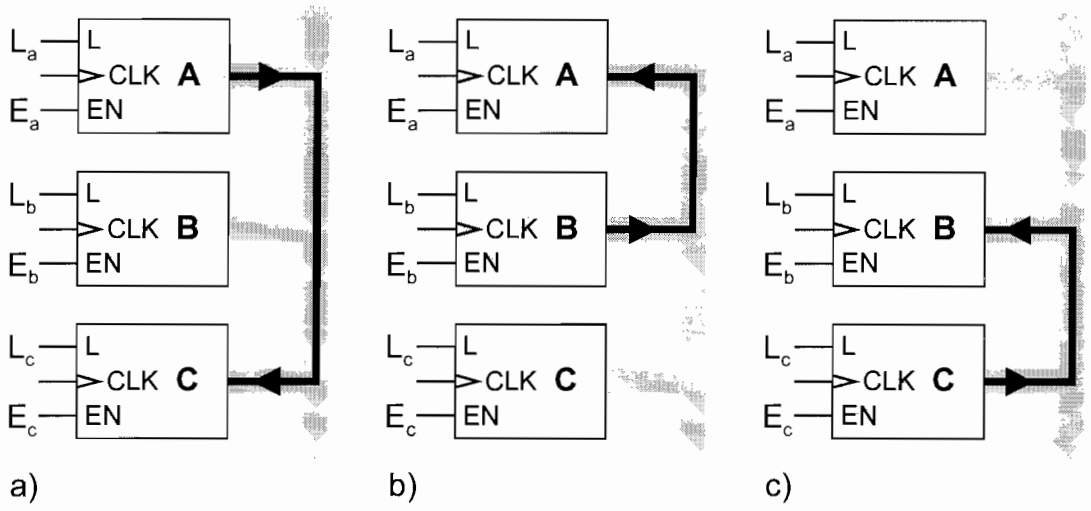



ABB. 2-18: Datenaustausch der Register A und B:
a) C := A, b) A := B, c) B := C

Diese schrittweise Datenübertragung führt letztendlich dazu, daß nach 3 Taktschritten die Register A und B ihren Inhalt getauscht haben, wobei allerdings der Inhalt von C verloren gegangen ist. Vermeiden ließe sich dieser Datenverlust nur, wenn A und B über getrennte Datenwege gekoppelt werden könnten, oder ein für solche Zwecke reserviertes Zwischenregister genutzt werden könnte. Wir werden später sehen, daß auch unter Zuhilfenahme eines externen Speichers (Stack) eine solche Zwischenspeicherung möglich ist.

Zur Steuerung dieses Vorgangs sollen die entsprechenden Steuersignale in Form eines Steuerworts CON = ($L_a$, $E_a$,..., $E_c$) angegeben werden:

| **CON** | **Steuersignale** | | | | | |
|---|---|---|---|---|---|---|
| | **$L_a$** | **$E_a$** | **$L_b$** | **$E_b$** | **$L_c$** | **$E_c$** |
| CON1 | 0 | 1 | 0 | 0 | 1 | 0 |
| CON2 | 1 | 0 | 0 | 1 | 0 | 0 |
| CON3 | 0 | 0 | 1 | 0 | 0 | 1 |

TABELLE 2-4: Steuerworte für Datenaustausch der Register A und B

Im ersten Schritt wird der Inhalt des Registers A auf den Datenbus freigegeben ($E_a$ =1) und mit der nächsten positiven Flanke des CLK-Signals in das Register C geladen ($L_c$=1). Dieser Vorgang wiederholt sich nun entsprechend in den Schritten 2 und 3.

Das zeitliche Geschehen (Timing) auf dem Datenbus und in den Registern wird durch folgendes Impulsdiagramm beschrieben:

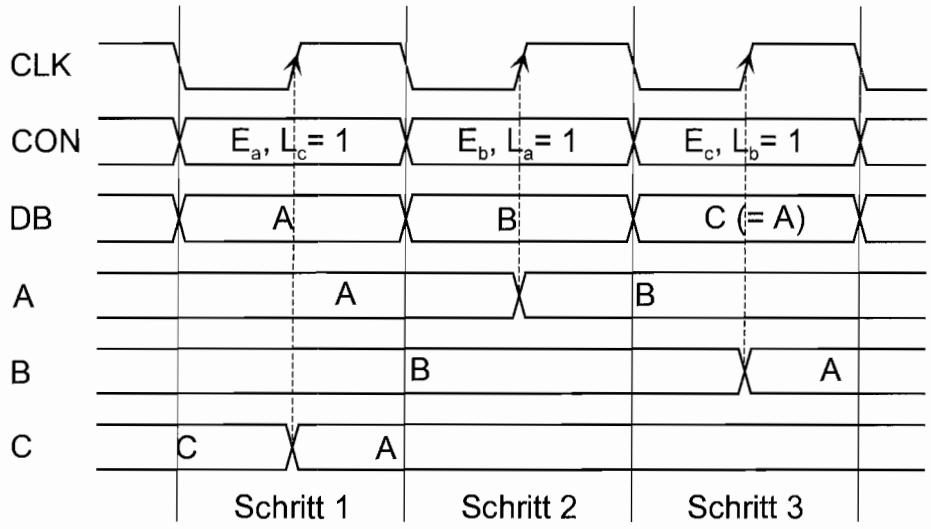



ABB. 2-19: Impulsdiagramm für das Timing des Datenaustauschs

Hervorzuheben ist bei diesem zeitlichen Ablauf, daß die Steuersignale phasenverschoben zu den Zustandsänderungen der Register erzeugt werden. Die Steuersignale müssen bereits früh genug vor dem Zustandwechsel der Register für stationäre Zustände an den Dateneingängen sorgen. Dies geschieht z. B. dadurch, daß die Steuersignale wie in dem obigen Impulsdiagramm eine halbe Taktperiode vor dem Zeitpunkt des Zustandswechsels aktiviert werden.